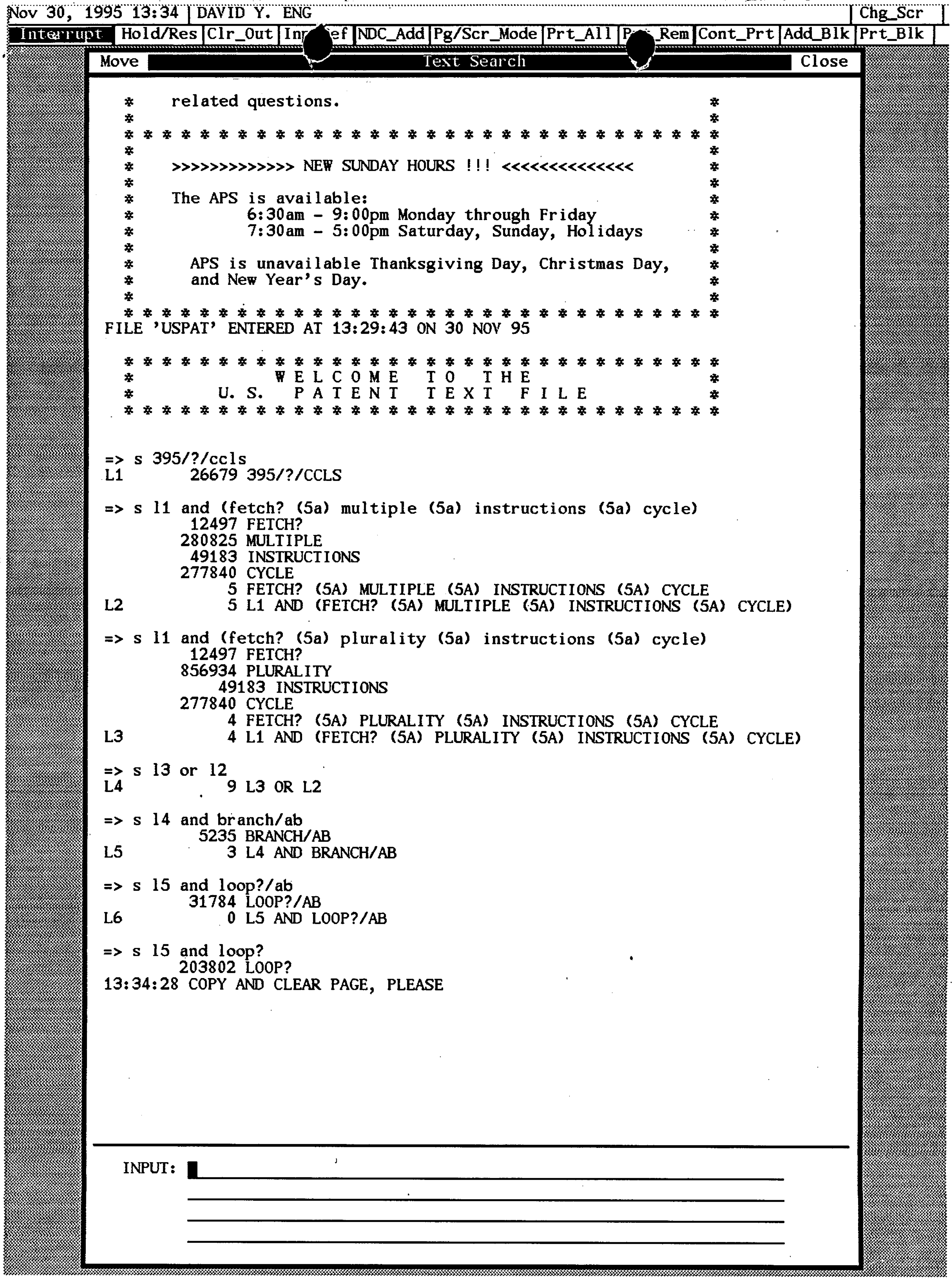
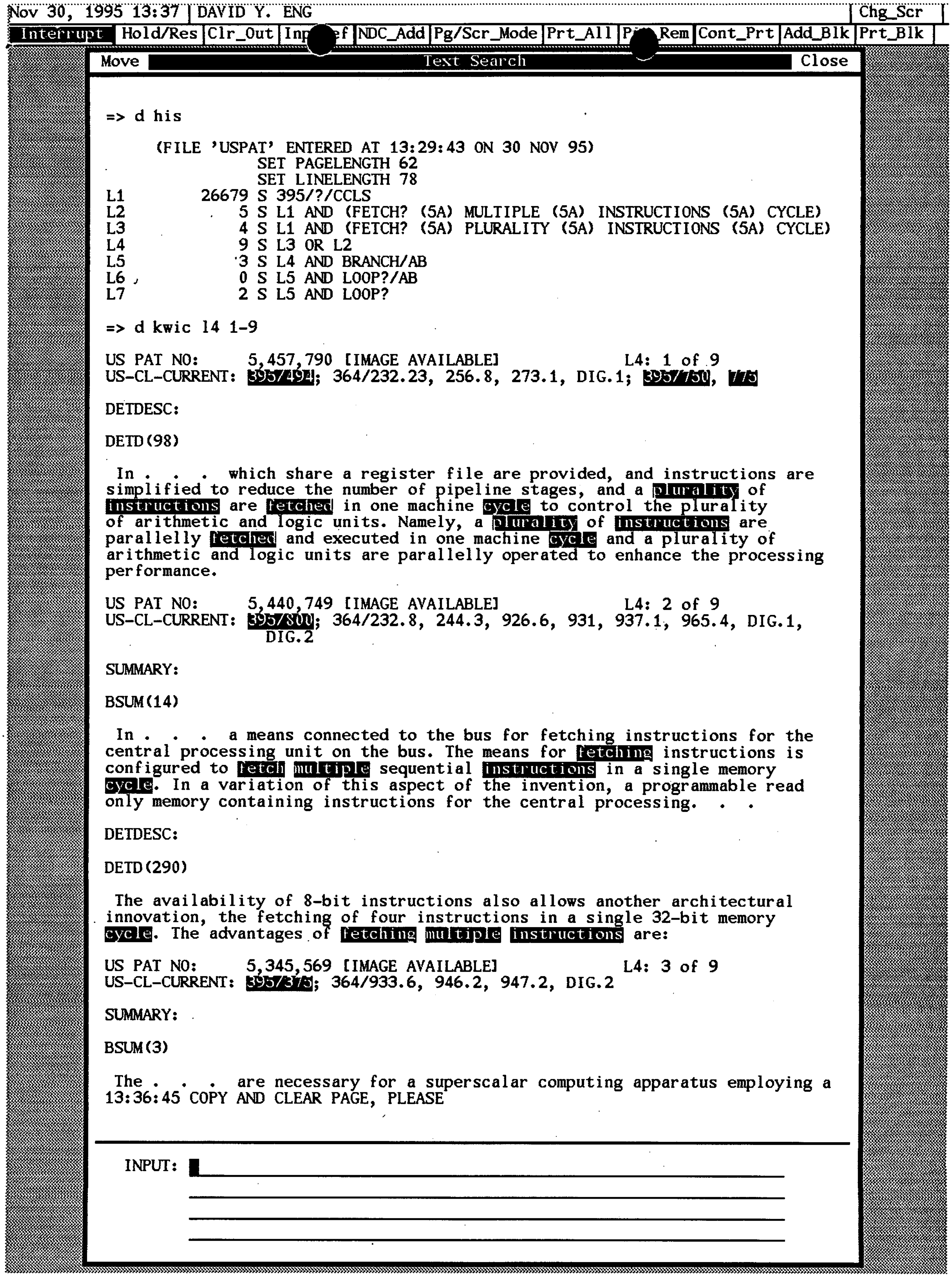


Nov 30, 1995 13:34 | DAVID Y. ENG

```
  *    related questions.                                          *
  *                                                                *
  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  *                                                                *
  *    >>>>>>>>>>>>>> NEW SUNDAY HOURS !!! <<<<<<<<<<<<<<          *
  *                                                                *
  *    The APS is available:                                       *
  *            6:30am - 9:00pm Monday through Friday               *
  *            7:30am - 5:00pm Saturday, Sunday, Holidays          *
  *                                                                *
  *      APS is unavailable Thanksgiving Day, Christmas Day,       *
  *      and New Year's Day.                                       *
  *                                                                *
  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
FILE 'USPAT' ENTERED AT 13:29:43 ON 30 NOV 95

  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  *             W E L C O M E   T O   T H E                        *
  *        U. S.   P A T E N T   T E X T   F I L E                 *
  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


=> s 395/?/ccls
L1        26679 395/?/CCLS

=> s l1 and (fetch? (5a) multiple (5a) instructions (5a) cycle)
          12497 FETCH?
         280825 MULTIPLE
          49183 INSTRUCTIONS
         277840 CYCLE
              5 FETCH? (5A) MULTIPLE (5A) INSTRUCTIONS (5A) CYCLE
L2            5 L1 AND (FETCH? (5A) MULTIPLE (5A) INSTRUCTIONS (5A) CYCLE)

=> s l1 and (fetch? (5a) plurality (5a) instructions (5a) cycle)
          12497 FETCH?
         856934 PLURALITY
             49183 INSTRUCTIONS
         277840 CYCLE
              4 FETCH? (5A) PLURALITY (5A) INSTRUCTIONS (5A) CYCLE
L3            4 L1 AND (FETCH? (5A) PLURALITY (5A) INSTRUCTIONS (5A) CYCLE)

=> s l3 or l2
L4            9 L3 OR L2

=> s l4 and branch/ab
           5235 BRANCH/AB
L5            3 L4 AND BRANCH/AB

=> s l5 and loop?/ab
          31784 LOOP?/AB
L6            0 L5 AND LOOP?/AB

=> s l5 and loop?
         203802 LOOP?
13:34:28 COPY AND CLEAR PAGE, PLEASE
```

```
Nov 30, 1995 13:35 | DAVID Y. ENG                                        | Chg_Scr |
Interrupt | Hold/Res | Clr_Out | In[illegible]ef | NDC_Add | Pg/Scr_Mode | Prt_All | [illegible]_Rem | Cont_Prt | Add_Blk | Prt_Blk
Move                              Text Search                              Close

L7            2 L5 AND LOOP?

=> d 17 kwic 1-2

US PAT NO:      5,203,002 [IMAGE AVAILABLE]               L7: 1 of 2
US-CL-CURRENT: 395/800; 364/231.8, 232.2, 244.8, 247.6, 254.1, DIG.1; 395/375

ABSTRACT:
An . . .  selects a next multi-instruction from a plurality of
multi-instructions thus allowing each multi-instruction, which may include a
plurality of different branch addresses, to execute in a single clock cycle.

SUMMARY:

BSUM(12)

 A . . .  of a multi-instruction may be performed during the access of
instruction operands. Since there are multiple instructions executed each
clock cycle, there are multiple next multi-instructions fetched from
the multiport memory in absolute parallel. One of these multi-instructions is
then selected to be executed next. This mechanism,. . .

SUMMARY:

BSUM(25)

 Now . . .  cycles to perform the required comparisons and assignments.
Multiple branches would also be required when implemented by means of a
loop. In contrast, this invention can perform the full fixed string
comparison operation in absolute parallel within one clock cycle. Both. . .

US PAT NO:      5,136,696 [IMAGE AVAILABLE]               L7: 2 of 2
US-CL-CURRENT: 395/375; 364/231.8, 243.41, 244.3, 247, 247.2, 247.5, 258.1,
                 261.3, 262.4, 262.8, 262.9, 263, 263.1, 275.8, DIG.1

ABSTRACT:
A . . .  the processing cycle immediately following the return prediction,
and normal program flow is resumed. The prediction cache memory also stores
branch instruction predictions and branch target addresses.

DETDESC:

DETD(69)

 In . . .  pair of counters, the normal instruction counter and an
interpreter counter. When the interpreter is entered, the normal counter is
looped back onto itself and does not increment during interpreter routines.
When the control is returned from the interpreter, the normal. . .

CLAIMS:

CLMS(27)

 27. Digital processing apparatus for executing stored program instructions
including single-cycle instructions and multicycle instructions during
multiple processing cycles comprising:
 an instruction fetching stage comprising
  instruction memory means for providing said program instructions in
   response to program addresses and for providing microinstructions of. .
.
13:34:54 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```



```
Nov 30, 1995 13:37 | DAVID Y. ENG                                    | Chg_Scr |
Interrupt | Hold/Res | Clr_Out | Inp  ef | NDC_Add | Pg/Scr_Mode | Prt_All | P   Rem | Cont_Prt | Add_Blk | Prt_Blk
Move                              Text Search                              Close
```

```
=> d his

     (FILE 'USPAT' ENTERED AT 13:29:43 ON 30 NOV 95)
                SET PAGELENGTH 62
                SET LINELENGTH 78
L1         26679 S 395/?/CCLS
L2             5 S L1 AND (FETCH? (5A) MULTIPLE (5A) INSTRUCTIONS (5A) CYCLE)
L3             4 S L1 AND (FETCH? (5A) PLURALITY (5A) INSTRUCTIONS (5A) CYCLE)
L4             9 S L3 OR L2
L5             3 S L4 AND BRANCH/AB
L6             0 S L5 AND LOOP?/AB
L7             2 S L5 AND LOOP?

=> d kwic l4 1-9

US PAT NO:      5,457,790 [IMAGE AVAILABLE]              L4: 1 of 9
US-CL-CURRENT: 395/494; 364/232.23, 256.8, 273.1, DIG.1; 395/750, 775

DETDESC:

DETD(98)

 In . . . which share a register file are provided, and instructions are
simplified to reduce the number of pipeline stages, and a plurality of
instructions are fetched in one machine cycle to control the plurality
of arithmetic and logic units. Namely, a plurality of instructions are
parallelly fetched and executed in one machine cycle and a plurality of
arithmetic and logic units are parallelly operated to enhance the processing
performance.

US PAT NO:      5,440,749 [IMAGE AVAILABLE]              L4: 2 of 9
US-CL-CURRENT: 395/800; 364/232.8, 244.3, 926.6, 931, 937.1, 965.4, DIG.1,
                DIG.2

SUMMARY:

BSUM(14)

 In . . . a means connected to the bus for fetching instructions for the
central processing unit on the bus. The means for fetching instructions is
configured to fetch multiple sequential instructions in a single memory
cycle. In a variation of this aspect of the invention, a programmable read
only memory containing instructions for the central processing. . .

DETDESC:

DETD(290)

 The availability of 8-bit instructions also allows another architectural
innovation, the fetching of four instructions in a single 32-bit memory
cycle. The advantages of fetching multiple instructions are:

US PAT NO:      5,345,569 [IMAGE AVAILABLE]              L4: 3 of 9
US-CL-CURRENT: 395/375; 364/933.6, 946.2, 947.2, DIG.2

SUMMARY:

BSUM(3)

 The . . . are necessary for a superscalar computing apparatus employing a
13:36:45 COPY AND CLEAR PAGE, PLEASE
```

```
  INPUT: █
```

```
Nov 30, 1995 13:37 | DAVID Y. ENG                                          | Chg_Scr
Interrupt | Hold/Res | Clr_Out | In[illegible]ef | NDC_Add | Pg/Scr_Mode | Prt_All | F[illegible]Rem | Cont_Prt | Add_Blk | Prt_Blk
Move                          Text Search                           Close

US PAT NO:      5,345,569 [IMAGE AVAILABLE]              L4: 3 of 9

BSUM(3)
pipelined instruction processing technique. Typically in such a computing
apparatus a fetch-batch of a plurality of instructions is processed
during each cycle. A design goal for such apparata is to dispatch and
execute multiple instructions per cycle, but impediments to reaching that.  .
.

US PAT NO:      5,317,718 [IMAGE AVAILABLE]              L4: 4 of 9
US-CL-CURRENT: 395/464; 364/243.45, 244.5, DIG.1; 395/449, 455, 496

DETDESC:

DETD(33)

 A .  .  .  take place in the time required to initiate a single level-two
cache reference. This is especially true in machines that fetch multiple
instructions per cycle from an instruction cache 18 and can concurrently
perform a load or store per cycle to a data cache 20..  .  .

US PAT NO:      5,261,066 [IMAGE AVAILABLE]              L4: 5 of 9
   US-CL-CURRENT: 395/449; 364/243.45, DIG.1; 395/455

DETDESC:

DETD(33)

 A .  .  .  take place in the time required to initiate a single level-two
cache reference. This is especially true in machines that fetch multiple
instructions per cycle from an instruction cache 18 and can concurrently
perform a load or store per cycle to a data cache 20..  .  .

US PAT NO:      5,203,002 [IMAGE AVAILABLE]              L4: 6 of 9
US-CL-CURRENT: 395/800; 364/231.8, 232.2, 244.8, 247.6, 254.1, DIG.1; 395/375

SUMMARY:

BSUM(12)

 A .  .  .  of a multi-instruction may be performed during the access of
instruction operands. Since there are multiple instructions executed each
clock cycle, there are multiple next multi-instructions fetched from
the multiport memory in absolute parallel. One of these multi-instructions is
then selected to be executed next. This mechanism,.  .  .

US PAT NO:      5,136,696 [IMAGE AVAILABLE]              L4: 7 of 9
US-CL-CURRENT: 395/375; 364/231.8, 243.41, 244.3, 247, 247.2, 247.5, 258.1,
                 261.3, 262.4, 262.8, 262.9, 263, 263.1, 275.8, DIG.1

CLAIMS:

CLMS(27)

 27. Digital processing apparatus for executing stored program instructions
including single-cycle instructions and multicycle instructions during
multiple processing cycles comprising:
 an instruction fetching stage comprising
  instruction memory means for providing said program instructions in
13:37:22 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

```
Nov 30, 1995 13:37 | DAVID Y. ENG                                          | Chg_Scr |
Interrupt | Hold/Res | Clr_Out | In[illegible]ef | NDC_Add | Pg/Scr_Mode | Prt_All | F[illegible]Rem | Cont_Prt | Add_Blk | Prt_Blk

Move                         Text Search                         Close

US PAT NO:      5,136,696 [IMAGE AVAILABLE]              L4: 7 of 9

CLMS(27)
   response to program addresses and for providing microinstructions of.  .
   .

US PAT NO:      5,134,701 [IMAGE AVAILABLE]              L4: 8 of 9
US-CL-CURRENT: 395/500; 364/264, 264.4, 267, 267.2, 921.8, 921.9, 925.6,
                928.4, 933.8, 935.53, 935.55, 938, 938.4, 941, 941.7, 949,
                949.1, 955, 955.3, 973, 976, 976.4, DIG.2; 395/183.14 .

DETDESC:

DETD(9)

 An example of a processor that retrieves a plurality of instructions on
every instruction fetch cycle is the Motorola Incorporated MC68020
Microprocessor. This device has an instruction repertoire based on sixteen
bit words. However, to improve.  .  .

US PAT NO:      5,127,091 [IMAGE AVAILABLE]              L4: 9 of 9
US-CL-CURRENT: 395/375; 364/228.6, 229, 229.2, 230.6, 231.8, 243, 243.4,
                243.41, 243.42, 261.3, 261.4, 261.5, 263, 263.1, 271.2,
                931.5, 938.1, 938.2, 948, 948.3, DIG.1; 395/800

CLAIMS:

CLMS(2)

 2. A data processing system according to claim 1 wherein said means for
storing a sequence of instructions includes means for fetching a
plurality of instructions during a cycle.

CLAIMS:

CLMS(8)

 8. A data processing system according to claim 7 wherein said means for
storing a sequence of instructions includes means for fetching a
plurality of instructions during a cycle.

CLAIMS:

CLMS(16)

 16. A data processing system according to claim 15 wherein said means for
storing a sequence of instructions includes means for fetching a
plurality of instructions during a cycle.

CLAIMS:

CLMS(22)

 22. A data processing system according to claim 21 wherein said means for
storing a sequence of instructions includes means for fetching a
plurality of instructions during a cycle.

CLAIMS:
13:37:42 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

```
Nov 30, 1995 13:37 | DAVID Y. ENG                                                    | Chg_Scr |
Interrupt | Hold/Res | Clr_Out | Inp[illegible]ef | NDC_Add | Pg/Scr_Mode | Prt_All | P[illegible]Rem | Cont_Prt | Add_Blk | Prt_Blk
Move                              Text Search                              Close

US PAT NO:      5,127,091 [IMAGE AVAILABLE]               L4: 9 of 9

CLMS(27)

 27. . . .
 determination of a branch instruction, for storing a sequence of branch
  target instructions in said storing means;
 means, connected to said fetching means, for dispatching a plurality of
  said fetched instructions in a cycle to a first processor for
  execution and for dispatching instructions in said sequence after said
  branch instruction to said first. . .

CLAIMS:

CLMS(32)

 32. . . .
 includes a branch instruction and in response to a determination of a branch
  instruction, storing a sequence of branch target instructions;
 dispatching a plurality of said fetched instructions in a cycle to a
  first processor for execution and dispatching instructions in said sequence
  after said branch instruction to said first processor. . .

=>

  INPUT:
```

United States Patent [19]

Iwamura et al.

[11] **Patent Number: 5,457,790**

[45] **Date of Patent: Oct. 10, 1995**

[54] **LOW POWER CONSUMPTION SEMICONDUCTOR INTEGRATED CIRCUIT DEVICE AND MICROPROCESSOR**

[75] Inventors: **Masahiro Iwamura; Shigeya Tanaka; Hideo Maejima, all of Hitachi; Tetsuo Nakano, Tokyo, all of Japan**

[73] Assignee: **Hitachi, Ltd., Tokyo, Japan**

[21] Appl. No.: **136,990**

[22] Filed: **Oct. 18, 1993**

Related U.S. Application Data

[63] Continuation of Ser. No. 973,576, Nov. 9, 1992, abandoned, which is a continuation of Ser. No. 627,847, Dec. 14, 1990, abandoned.

[30] **Foreign Application Priority Data**

Dec. 15, 1989	[JP]	Japan	1-324928
Aug. 3, 1990	[JP]	Japan	2-205006

[51] **Int. Cl.**^{6} **G06F 1/32**

[52] **U.S. Cl.** 395/494; 395/750; 395/775; 364/232.23; 364/256.8; 364/273.1; 364/DIG. 1

[58] **Field of Search** .... 395/425, 750, 395/775, 375, 800; 364/DIG. 1, 273.1, 256.8, 232.23, DIG. 2, 948.8, 961.3, 931.55, 707; 365/227

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,736,569	5/1973	Bouricius et al.	395/750
4,137,563	1/1979	Tsunoda	395/550
4,236,205	11/1980	Kindseth et al.	395/425
4,961,172	10/1990	Shubat et al.	365/230.06
5,060,188	10/1991	Zulian et al.	395/425

FOREIGN PATENT DOCUMENTS

2825770A1	1/1980	Germany	G06F 1/00
61-45354	3/1986	Japan	G06F 15/06

OTHER PUBLICATIONS

Nikke: Electronics, "Special Edition: Electronics of the 1990's", pp. 191–200, Nov. 27, 1989.

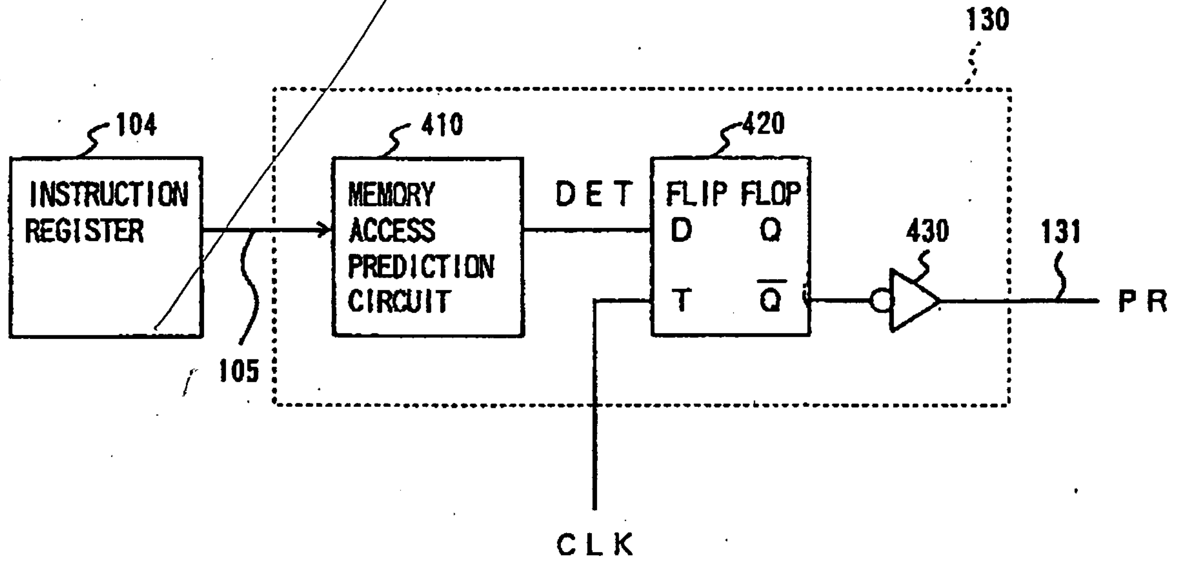
Von Karl Reiss, "Integrierte Digitalbausteine", Siemens Aktiengesellschaft, 1970, pp. 214–219.

Primary Examiner—Jack B. Harvey
Assistant Examiner—Glenn A. Auve
Attorney, Agent, or Firm—Antonelli, Terry, Stout & Kraus

[57] **ABSTRACT**

In semiconductor integrated circuit device and microprocessor including at least one functional circuit block, the start of operation of the functional circuit block is detected prior to the start of operation, the functional circuit block for which the start of operation has been detected is activated prior to the start of operation and inactivated after the termination of operation.

19 Claims, 24 Drawing Sheets



United States Patent [19]

Moore et al.

[11] Patent Number: 5,440,749
[45] Date of Patent: Aug. 8, 1995

[54] HIGH PERFORMANCE, LOW COST MICROPROCESSOR ARCHITECTURE

[75] Inventors: Charles H. Moore, Woodside; Russell H. Fish, III, Mt. View, both of Calif.

[73] Assignee: Nanotronics Corporation, Eagle Point, Oreg.

[21] Appl. No.: 389,334

[22] Filed: Aug. 3, 1989

[51] Int. Cl.⁶ G06F 9/22
[52] U.S. Cl. 395/800; 364/931; 364/925.6; 364/937.1; 364/965.4; 364/232.8; 364/244.3
[58] Field of Search 395/425, 725, 775, 800

[56] References Cited

U.S. PATENT DOCUMENTS

3,603,934	9/1971	Heath	364/DIG. 1
4,003,033	1/1977	O'Keefe et al.	364/200
4,037,090	7/1977	Raymond	364/200
4,042,972	8/1977	Grunes et al.	364/200
4,050,058	9/1977	Garlic	395/800
4,067,059	1/1978	Derchak	364/DIG. 1
4,079,455	3/1978	Ozga	395/800
4,110,822	8/1978	Porter	364/200
4,125,871	11/1978	Martin	364/DIG. 2
4,128,873	12/1978	Lamiaux	364/200
4,255,785	3/1981	Chamberlin	395/375
4,354,228	10/1982	Moore et al.	364/200
4,376,977	3/1983	Brunshorst	364/DIG. 1
4,382,279	5/1983	Mgon	364/200
4,403,303	9/1983	Howes et al.	364/900
4,450,519	5/1984	Guttag et al.	364/200
4,463,421	7/1984	Laws	395/325
4,538,239	8/1985	Magar	364/759
4,541,045	9/1985	Kromer	395/375
4,562,537	12/1985	Barnett et al.	395/375
4,577,282	3/1986	Caudel et al.	395/800
4,607,332	8/1986	Goldberg	364/900
4,626,988	12/1986	George et al.	364/200
4,649,471	3/1987	Briggs	395/325
4,665,495	5/1987	Thaden	345/185
4,709,329	11/1987	Hecker	395/275
4,713,749	12/1987	Magar et al.	395/375
4,714,994	12/1987	Oklobdzija et al.	395/375
4,720,812	1/1988	Kao et al.	395/700
4,772,888	9/1988	Kimura	340/825.5
4,777,591	10/1988	Chang et al.	395/800
4,787,032	11/1988	Culley et al.	364/200
4,803,621	2/1989	Kelly	395/400
4,860,198	8/1989	Takenaka	364/DIG. 1
4,870,562	9/1989	Kimoto	364/DIG. 1
4,931,986	6/1990	Daniel et al.	395/550
5,036,460	7/1991	Takahira	395/425
5,070,451	12/1991	Moore et al.	395/375
5,127,091	6/1992	Bonfarah	395/375

OTHER PUBLICATIONS

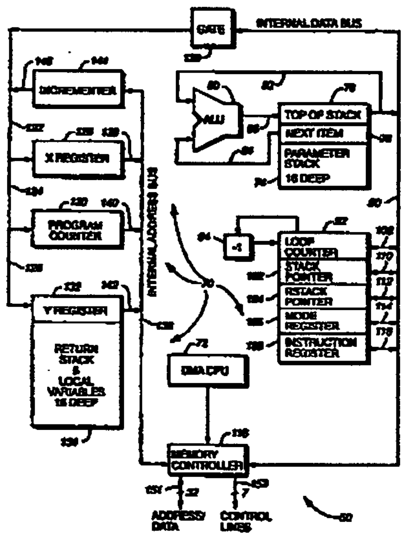
Intel 80386 Programmer's Reference Manual, 1986.

Primary Examiner—David Y. Eng
Attorney, Agent, or Firm—Cooley Godward Castro Huddleson & Tatum

[57] ABSTRACT

A microprocessor (50) includes a main central processing unit (CPU) (70) and a separate direct memory access (DMA) CPU (72) in a single integrated circuit making up the microprocessor (50). The main CPU (70) has a first 16 deep push down tack (74), which has a to item register (76) and a next item register (78), respectively connected to provide inputs to an arithmetic logic unit (ALU) (80) by lines (82) and (84). An output of the ALU (80) is connected to the top item register at (82) is also connected by line (88) to an internal data bus (90). CPU (70) is pipeline free. The simplified CPU (70) requires fewer transistors to implement than pipelined architectures, yet produces performance which matches or exceeds existing techniques. The DMA CPU (72) provides inputs to the memory controller (118) on line (148). The memory controller (118) is connected to a RAM by address/data bus (150) and control lines (152). The DMA CPU (72) enables the CPU (70) to execute instructions four times faster than the RAM speed by fetching four instructions in a single memory cycle.

29 Claims, 19 Drawing Sheets



United States Patent [19]

Tran

[11] Patent Number: 5,345,569
[45] Date of Patent: Sep. 6, 1994

[54] APPARATUS AND METHOD FOR RESOLVING DEPENDENCIES AMONG A PLURALITY OF INSTRUCTIONS WITHIN A STORAGE DEVICE

[75] Inventor: Thang M. Tran, Austin, Tex.

[73] Assignee: Advanced Micro Devices, Inc., Sunnyvale, Calif.

[21] Appl. No.: 764,155

[22] Filed: Sep. 20, 1991

[51] Int. Cl.^{5} G06F 9/06
[52] U.S. Cl. 395/375; 364/DIG. 2; 364/933.6; 364/947.2; 364/946.2
[58] Field of Search .... 395/375; 364/DIG. 1 MS File, DIG. 2 MS File

[56] References Cited

U.S. PATENT DOCUMENTS

4,714,994	12/1987	Oklobdzija et al.	395/375
4,879,646	11/1989	Iwasaki et al.	395/375
5,226,126	7/1993	McFarland et al.	395/375

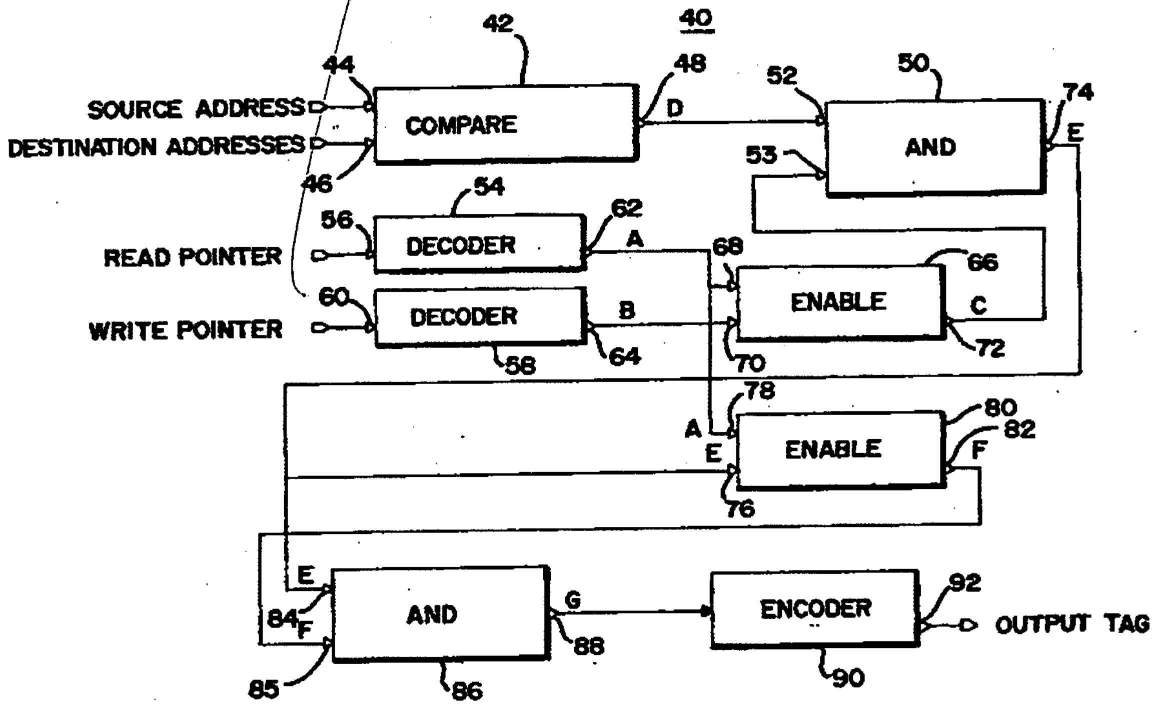
Primary Examiner—Thomas M. Heckler
Attorney, Agent, or Firm—Foley & Lardner

[57] ABSTRACT

An apparatus and method for resolving data dependencies among a plurality of instructions within a storage device, such as a reorder buffer in a superscalar computing apparatus employing pipeline instruction processing. The storage device has a read pointer, indicating a most recently-stored instruction and has a write pointer, indicating a first-stored instruction of the plurality of instructions within the storage device.

A compare-hit circuit generates a compare-hit signal upon each concurrence of the respective source indicator in a next-to-be-dispatched instruction with the destination indicator of an earlier-stored instruction within the storage device; a first enable circuit generates a first enable signal for a first packet of instructions defined by the read pointer and the write pointer; a first comparing circuit generates a hit-enable signal for each concurrence of the compare-hit signal and the first enable signal; a second enable circuit generates a second enable signal for a second packet of instructions defined by the read pointer and the hit-enable signal; and a second comparing circuit generates the output signal for each concurrence of the second enable signal and the hit-enable signal.

12 Claims, 4 Drawing Sheets



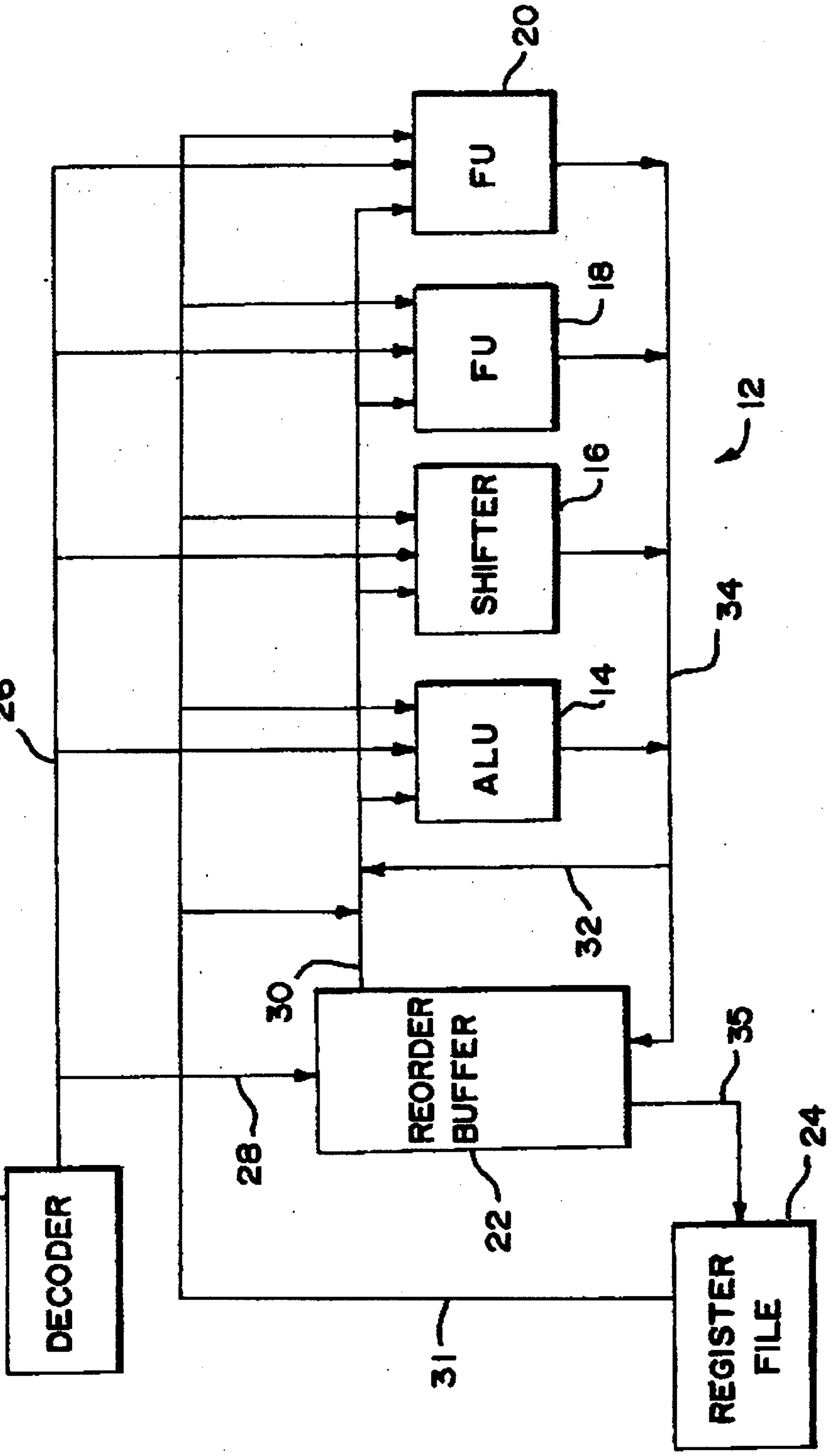
Nov 30, 1995 13:39 | DAVID Y. ENG | Chg_Scr

Interrupt | Tag | Del | Skip | More | P[illegible] Loc | Sec_dis | Dis_col | Set_ndcm | R[illegible]ndcm | Aux

==> Patent Retrieval Window

Doc. 05345569 OR 395/375 XR 364/933.6 + Sec. D Page 2 of 11 Doc 3 of 9

FIG. I



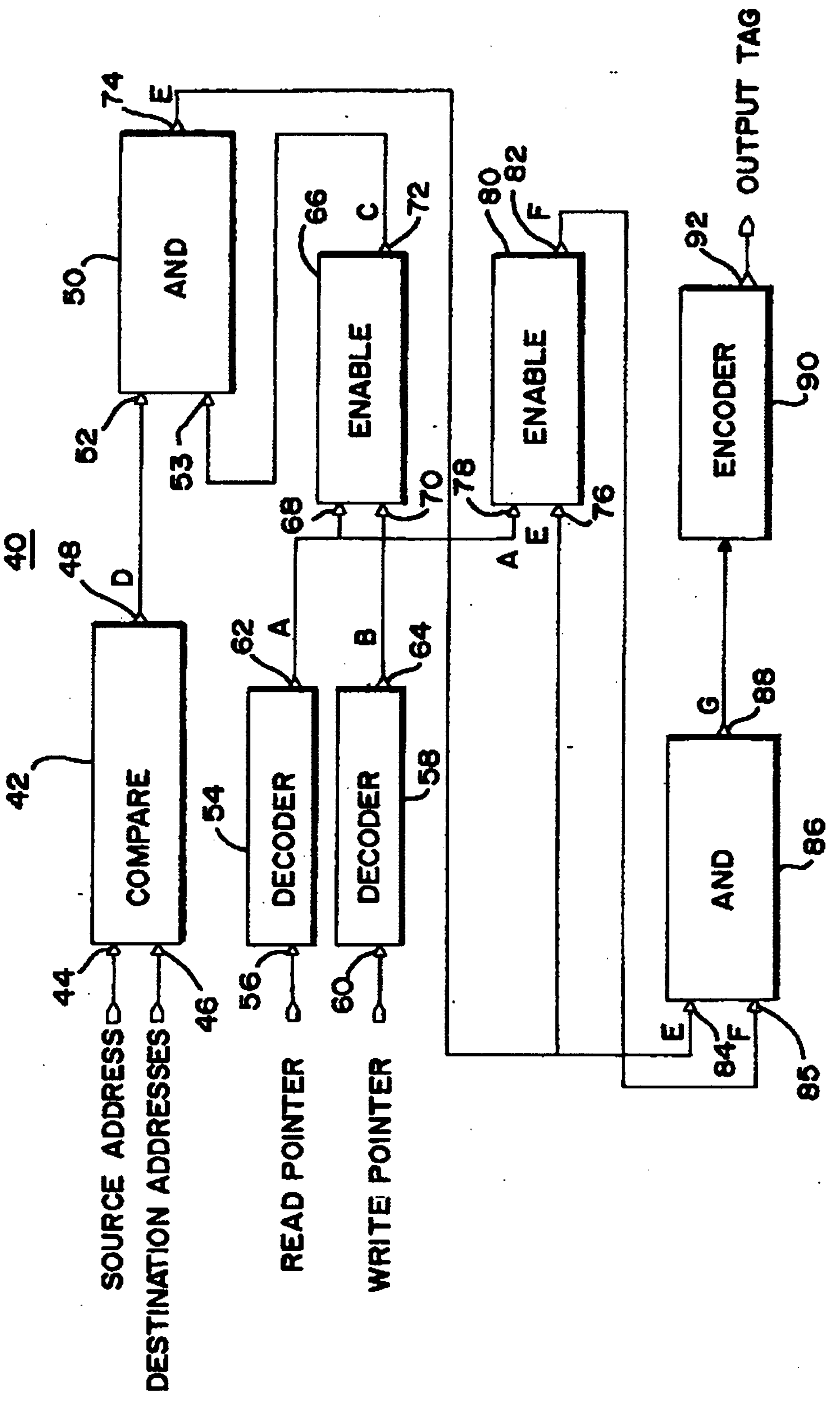

40
SOURCE ADDRESS
DESTINATION ADDRESSES
44
46
COMPARE
42
48
D
READ POINTER
WRITE POINTER
56
60
DECODER
DECODER
54
58
62
64
A
B
AND
50
52
53
74
E
ENABLE
66
68
70
72
C
ENABLE
76
78
80
82
A
E
F
AND
86
84
85
E
F
88
G
ENCODER
90
92
OUTPUT TAG

FIG. 2

US005317718A

United States Patent [19]

Jouppi

[11] **Patent Number: 5,317,718**
[45] **Date of Patent: May 31, 1994**

[54] **DATA PROCESSING SYSTEM AND METHOD WITH PREFETCH BUFFERS**

[75] Inventor: **Norman P. Jouppi**, Palo Alto, Calif.
[73] Assignee: **Digital Equipment Corporation**, Maynard, Mass.

[21] Appl. No.: **10,446**
[22] Filed: **Jan. 25, 1993**

Related U.S. Application Data

[63] Continuation of Ser. No. 500,062, Mar. 27, 1990, abandoned.

[51] Int. Cl.^{5} G06F 12/08
[52] U.S. Cl. 395/425; 364/DIG. 1; 364/243.45; 364/244.5
[58] Field of Search .... 395/400, 425; 364/200 MS File, 900 MS File

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,938,097	2/1976	Niguette, III	340/172.5
4,458,310	7/1984	Chang	364/200
4,464,717	8/1984	Keeley et al.	395/425
4,467,414	8/1984	Akagi et al.	364/200
4,823,259	4/1989	Aichelmann, Jr. et al.	395/425
4,847,753	7/1989	Matsuo et al.	
4,897,783	1/1990	Nay	364/200
4,926,323	5/1990	Baror et al.	
4,942,520	7/1990	Langendorf	395/425
4,974,156	11/1990	Harding et al.	395/425
5,023,776	6/1991	Gregor	395/425
5,101,341	3/1992	Circello et al.	395/375
5,136,697	8/1992	Johnson	395/375
5,148,536	9/1992	Witek et al.	395/425
5,163,140	11/1992	Stiles et al.	395/425
5,214,765	5/1993	Jensen	395/425

FOREIGN PATENT DOCUMENTS

2193356	2/1988	United Kingdom .

OTHER PUBLICATIONS

Liu, L., "Increasing Hit Ratios in Second Level Caches and Reducing the Size of Second Level Storage" IBM Technical Disclosure Bulletin: vol. 27, No. 1A, Jun. 1984.

Maytal, B., et al; "Design Considerations for a General-Purpose Microprocessor"; Computer, vol. 22, No. 1, Jan. 1989.

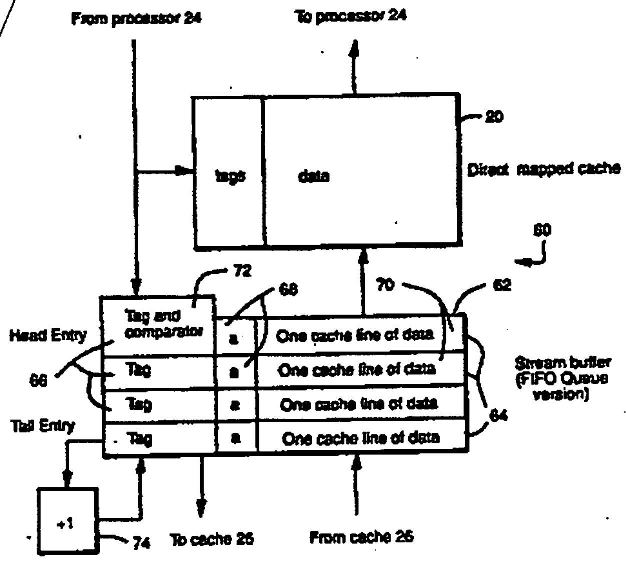
Jouppi, Norman, "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers", Computer Architectur News: vol. 18, No. 2, Jun. 1990.

Primary Examiner—Joseph L. Dixon
Assistant Examiner—Hiep T. Nguyen
Attorney, Agent, or Firm—Flehr, Hohbach, Test, Albritton & Herbert

[57] **ABSTRACT**

A memory system (10) utilizes miss caching by incorporating a small fully-associative miss cache (42) between a cache (18 or 20) and second-level cache (26). Misses in the cache (18 or 20) that hit in the miss cache have only a one cycle miss penalty, as opposed to a many cycle miss penalty without the miss cache (42). Victim caching is an improvement to miss caching that loads a small, fully associative cache (52) with the victim of a miss and not the requested line. Small victim caches (52) of 1 to 4 entries are even more effective at removing conflict misses than miss caching. Stream buffers (62) prefetch cache lines starting at a cache miss address. The prefetched data is placed in the buffer (62) and not in the cache (18 or 20). Stream buffers (62) are useful in removing capacity and compulsory cache misses, as well as some instruction cache misses. Stream buffers (62) are more effective than previously investigated prefetch techniques when the next slower level in the memory hierarchy is pipelined. An extension to the basic stream buffer, called multi-way stream buffers (62), is useful for prefetching along multiple intertwined data reference streams.

20 Claims, 21 Drawing Sheets



United States Patent [19]

Jouppi et al.

[11] **Patent Number: 5,261,066**

[45] **Date of Patent: Nov. 9, 1993**

[54] DATA PROCESSING SYSTEM AND METHOD WITH SMALL FULLY-ASSOCIATIVE CACHE AND PREFETCH BUFFERS

[75] Inventors: **Norman P. Jouppi; Alan Eustace,** both of Palo Alto, Calif.

[73] Assignee: **Digital Equipment Corporation,** Maynard, Mass.

[21] Appl. No.: **499,958**

[22] Filed: **Mar. 27, 1990**

[51] Int. Cl.^{5} **G06F 12/00; G06F 13/00; G06F 12/08**

[52] U.S. Cl. **395/425; 364/DIG. 1; 364/243.45**

[58] Field of Search .... **364/DIG. 1, DIG. 2, 364/243, 243.45, 243.7, 964, 343, 964.22, 964.23, 964.6, 964.5**

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,464,712	8/1984	Fletcher	395/425
4,783,736	11/1988	Ziegler et al.	364/200
4,794,521	12/1988	Ziegler et al.	364/200
4,797,814	1/1989	Brenza	364/200
4,811,209	3/1989	Rubenstein	364/200
4,853,846	8/1989	Johnson et al.	364/200
4,888,679	12/1989	Fossum et al.	395/800
4,928,225	5/1990	McCarthy et al.	395/425
4,953,073	8/1990	Moussouris et al.	364/200
4,969,122	11/1990	Jensen	365/49
5,027,270	6/1991	Riordan et al.	364/200
5,155,832	10/1992	Hunt	395/425

OTHER PUBLICATIONS

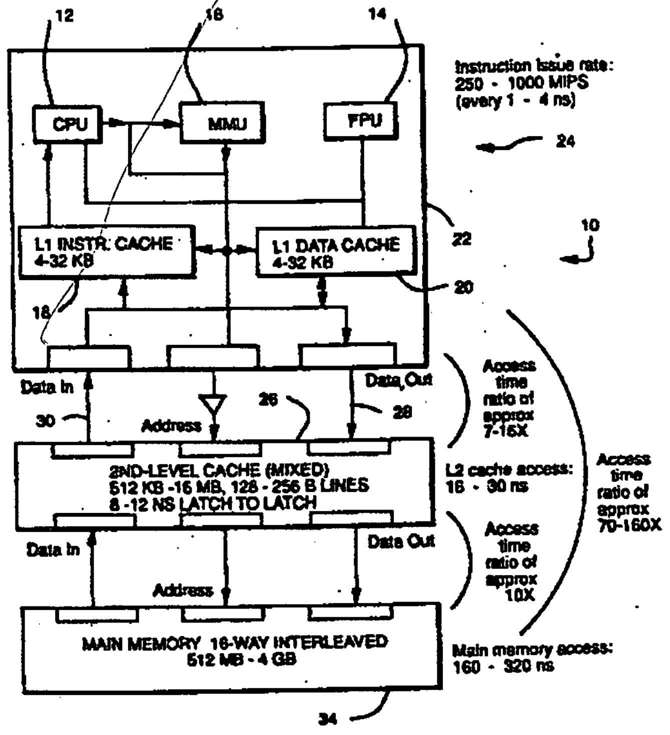
Smith, "Cache Memories," 14 *Computing Surveys* 473–530 (Sep. 1982).

Primary Examiner—Alyssa H. Bowler
Attorney, Agent, or Firm—Flehr, Hohbach, Test, Albritton & Herbert

[57] **ABSTRACT**

A memory system (10) utilizes miss caching by incorporating a small fully-associative miss cache (42) between a cache (18 or 20) and second-level cache (26). misses in the cache (18 or 20) that hit in the miss cache have only a one cycle miss penalty, as opposed to a many cycle miss penalty without the miss cache (42). Victim caching is an improvement to miss caching that loads a small, fully associative cache (52) with the victim of a miss and not the requested line. Small victim caches (52) of 1 to 4 entries are even more effective at removing conflict misses than miss caching. Stream buffers (62) prefetch cache lines starting at a cache miss address. The prefetched data is placed in the buffer (62) and not in the cache (18 or 20). Stream buffers (62) are useful in removing capacity and compulsory cache misses, as well as some instruction cache misses. Stream buffers (62) are more effective than previously investigated prefetch techniques when the next slower level in the memory hierarchy is pipelined. An extension to the basic stream buffer, called multi-way stream buffers (62), is useful for prefetching along multiple intertwined data reference streams.

16 Claims, 21 Drawing Sheets


12
18
14
CPU
MMU
FPU
Instruction issue rate: 250 - 1000 MIPS (every 1 - 4 ns)
24
22
10
L1 INSTR. CACHE 4-32 KB
L1 DATA CACHE 4-32 KB
20
18
Data In
Data Out
30
Address
26
28
Access time ratio of approx 7-16X
2ND-LEVEL CACHE (MIXED) 512 KB -16 MB, 128 - 256 B LINES 8 -12 NS LATCH TO LATCH
L2 cache access: 16 - 30 ns
Access time ratio of approx 70-160X
Data In
Data Out
Address
Access time ratio of approx 10X
MAIN MEMORY 16-WAY INTERLEAVED 512 MB - 4 GB
Main memory access: 160 - 320 ns
34

```
Nov 30, 1995 13:40 | DAVID Y. ENG                                                   | Chg_Scr |
 Interrupt | Tag|Del |Skip | More |Pr  Loc |Sec_dis|Dis_col | Set_ndcm|Re  ndcm|Aux
==>                                           Patent Retrieval Window
 Doc. 05203002     OR 395/800          XR 364/231.8    +  Sec. F     Page      1 of    15  Doc      6 of      9
```

US005203002A

United States Patent [19]

Wetzel

[11] **Patent Number: 5,203,002**

[45] **Date of Patent: Apr. 13, 1993**

[54] **SYSTEM WITH A MULTIPORT MEMORY AND N PROCESSING UNITS FOR CONCURRENTLY/INDIVIDUALLY EXECUTING 2N-MULTIINSTRUCTION-WORDS AT FIRST/SECOND TRANSITIONS OF A SINGLE CLOCK CYCLE**

[76] Inventor: Glen F. Wetzel, 1682 El Cerrito Ct., San Luis Obispo, Calif.

[21] Appl. No.: 457,515

[22] Filed: Dec. 27, 1989

[51] Int. Cl.^{5} G06F 9/38; G06F 9/30

[52] U.S. Cl. 395/800; 364/DIG. 1; 364/231.8; 364/232.2; 364/244.8; 364/247.6; 364/254.1; 395/375

[58] Field of Search ... 364/200 MS File, 900 MS File; 395/375, 800

[56] **References Cited**

U.S. PATENT DOCUMENTS

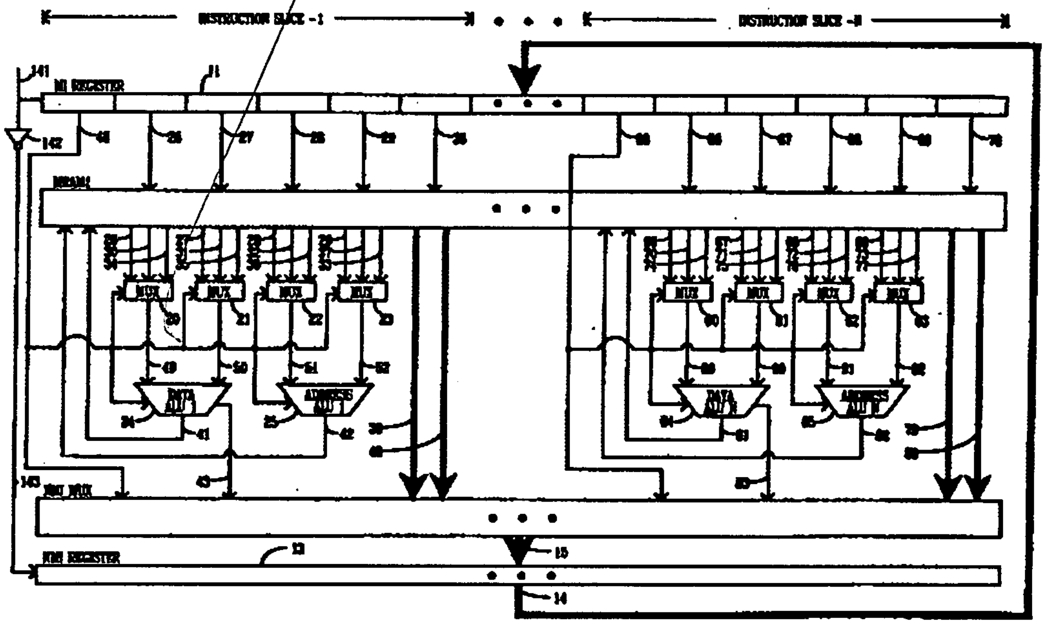
3,969,702	7/1976	Tessera	364/200
4,050,058	9/1977	Garlic	364/200
4,228,498	10/1980	Moshier	364/200
4,295,193	10/1981	Pomerene	364/200
4,310,879	1/1982	Pandeya	364/200
4,314,333	2/1982	Shibayama et al.	364/200
4,400,768	8/1983	Tomlinson	364/200
4,435,756	3/1984	Potash	364/200
4,476,525	10/1984	Ishii	364/200
4,507,728	3/1985	Sakamoto et al.	364/200
4,574,348	3/1986	Scallon	364/200
4,594,655	6/1986	Hao et al.	364/200
4,613,953	9/1986	Bush et al.	364/900
4,626,989	12/1986	Torii	364/200
4,636,942	1/1987	Chen et al.	364/200
4,654,788	3/1987	Boudreau et al.	395/425
4,709,327	11/1987	Hillis et al.	364/200
4,712,175	12/1987	Torii et al.	364/200
4,734,852	3/1988	Johnson et al.	364/200
4,740,894	4/1988	Lyon	364/200
4,752,873	1/1988	Shonai et al.	364/200
4,766,566	8/1988	Chuang	364/900
4,774,654	9/1988	Pomerene et al.	364/200
4,809,169	2/1989	Sarti et al.	395/800
4,819,155	4/1989	Wulf et al.	364/200
4,833,599	5/1989	Colwell et al.	364/200
4,837,676	6/1989	Rosman	364/200
4,847,755	7/1989	Morrison et al.	364/200
4,855,904	8/1989	Daberkow et al.	364/200
4,942,525	7/1990	Shintani et al.	364/200
4,954,947	9/1990	Kuriyama et al.	364/200
4,967,339	12/1990	Fukumaru et al.	364/200
4,989,140	1/1991	Nishimukai et al.	364/200

Primary Examiner—Thomas C. Lee
Assistant Examiner—Krisna Lim

[57] **ABSTRACT**

An improved digital processor mechanism capable of executing a plurality of instructions in absolute parallel. Instructions that execute in parallel are grouped into a multi-instruction word. The processor incorporates a multiport memory for storing multi-instructions, addresses and data, and a plurality of arithmetic and logic units to compute both the write address and write data for each instruction. The multiport memory allows a plurality of instruction operands and a plurality of multi-instructions to be fetched in parallel. In addition, the multiport memory allows a plurality of computer data to be written in parallel. A priority instruction multiplexer selects a next multi-instruction from a plurality of multi-instructions thus allowing each multi-instruction, which may include a plurality of different branch addresses, to execute in a single clock cycle.

18 Claims, 3 Drawing Sheets



United States Patent [19]

Beckwith et al.

[11] **Patent Number: 5,136,696**

[45] **Date of Patent: Aug. 4, 1992**

[54] **HIGH-PERFORMANCE PIPELINED CENTRAL PROCESSOR FOR PREDICTING THE OCCURRENCE OF EXECUTING SINGLE-CYCLE INSTRUCTIONS AND MULTICYCLE INSTRUCTIONS**

[75] Inventors: **Robert F. Beckwith**, Framingham; **Neil J. Johnson**, Burlington; **Suren Irukulla**, Holliston; **Steven Schwartz**, Sudbury; **Nihar Mohapatra**, Milford, all of Mass.

[73] Assignee: **Prime Computer, Inc.**, Framingham, Mass.

[21] Appl. No.: **211,977**

[22] Filed: **Jun. 27, 1988**

[51] Int. Cl.^{5} G06F 9/22

[52] U.S. Cl. 395/375; 364/262.8; 364/262.4; 364/275.8; 364/263.1; 364/DIG. 1

[58] Field of Search ... 364/200 MS File, 900 MS File

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,179,731	12/1979	Yamazaki	364/200
4,200,927	4/1980	Hughes	364/200
4,370,711	1/1983	Smith	364/200
4,399,507	8/1983	Cosgrove	364/200
4,454,580	6/1984	Page	364/20
4,477,872	10/1984	Losq	364/200
4,494,187	1/1985	Simpson	364/200
4,494,188	1/1985	Nakane	364/200
4,498,136	2/1985	Sproul, III	364/200
4,583,162	4/1986	Prill	364/200
4,679,141	7/1987	Pomerene	364/200
4,750,112	6/1988	Jones	364/200
4,760,519	7/1988	Papworth	364/200
4,764,861	8/1988	Shibuya	364/200
4,777,587	10/1988	Case	364/200
4,819,234	4/1989	Huber	364/200
4,847,753	7/1989	Matsuo	364/200
4,853,840	8/1989	Shibuya	364/200
4,860,197	8/1989	Langendorf	364/200
4,860,199	8/1989	Langendorf	364/200
4,872,109	10/1989	Horst	364/200
4,942,520	7/1990	Langendorf	364/200

OTHER PUBLICATIONS

A. Bandyopadhyay, "Combining both Micro-code & Hardwired Cont. in RISC", Comp. Arch. News, Sep. 1987, pp. 11-15.
D. S. Coutant et al, "Compilers for the New Generation of H-P Computers", H-P Journal, Jan. 1986, pp. 4-19.
R. D. Bernal, "Putting RISC Eff. to Work in CISC Arch.", VLSI Systems Des., Sep. 1987, pp. 46-51.
J. C. Circello, "Des. of the Edge 1 Supermini-comp.", VLSI Systems Des., May 1986, pp. 22-28.
J. Cho et al, "The Memory Arch. & the Cache & Mem. Mgmt. Unit . . . ", U. of Cal. Report #UCB/CSD 86/289, p. 5.
"4.2.2 Branch Target Cache", author and date unknown.

Primary Examiner—Thomas C. Lee
Assistant Examiner—Eric Coleman
Attorney, Agent, or Firm—Wolf, Greenfield & Sacks

[57] **ABSTRACT**

A pipelined central processor capable of executing both single-cycle instructions and multicycle instructions is provided. An instruction fetch stage of the processor includes an instruction cache memory and a prediction cache memory that are commonly addressed by a program counter register. The instruction cache memory stores instructions of a program being executed and microinstructions of a multicycle instruction interpreter. The prediction cache memory stores interpreter call predictions and interpreter entry addresses at the addresses of the multicycle intructions. When a call prediction occurs, the entry address of the instruction interpreter is loaded into the program counter register on the processing cycle immediately following the call prediction, and a return address is pushed onto a stack. The microinstructions of the interpreter are fetched sequentially from the instruction cache memory. When the interpreter is completed, the prediction cache memory makes a return prediction. The return address is transferred from the stack to the program counter register on the processing cycle immediately following the return prediction, and normal program flow is resumed. The prediction cache memory also stores branch instruction predictions and branch target addresses.

29 Claims, 8 Drawing Sheets

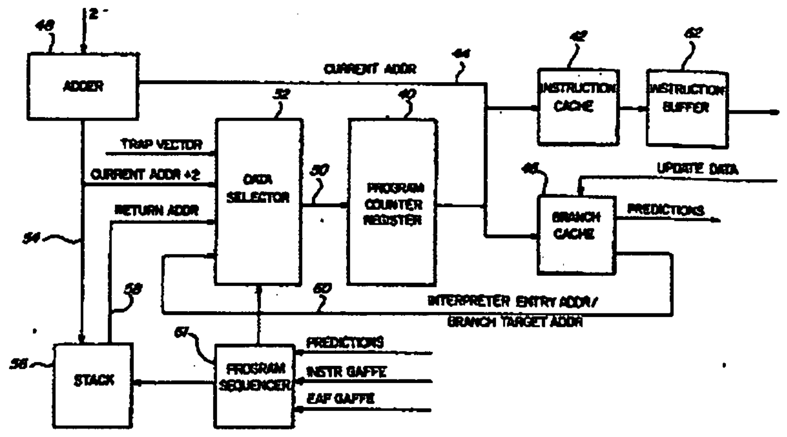
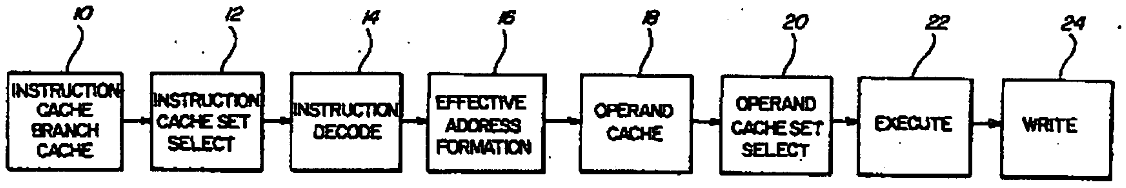


Fig.1


10
INSTRUCTION CACHE BRANCH CACHE
12
INSTRUCTION CACHE SET SELECT
14
INSTRUCTION DECODE
16
EFFECTIVE ADDRESS FORMATION
18
OPERAND CACHE
20
OPERAND CACHE SET SELECT
22
EXECUTE
24
WRITE

United States Patent [19]

Mueller et al.

[11] **Patent Number: 5,134,701**

[45] **Date of Patent: Jul. 28, 1992**

[54] **TEST APPARATUS PERFORMING RUNTIME REPLACEMENT OF PROGRAM INSTRUCTIONS WITH BREAKPOINT INSTRUCTIONS FOR PROCESSOR HAVING MULTIPLE INSTRUCTION FETCH CAPABILITIES**

[75] Inventors: **David C. Mueller; Steven R. Williams; Nabil M. Abu-Jbara**, all of Colorado Springs, Colo.

[73] Assignee: **Hewlett-Packard Co.**, Palo Alto, Calif.

[21] Appl. No.: **310,153**

[22] Filed: **Feb. 10, 1989**

[51] Int. Cl.^{5} **G06F 11/30**

[52] U.S. Cl. **395/500**; 364/264; 364/264.4; 364/267.2; 364/267; 364/DIG. 2; 371/19

[58] Field of Search ... 364/200 MS File, 900 MS File, 364/DIG. 2; 371/19; 395/500, 775, 725, 375

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,509,541	4/1970	Gordon	364/200
3,904,860	9/1975	Huber	364/200
4,080,650	3/1978	Beckett	364/200
4,126,893	11/1978	Cronshaw .	
4,176,394	11/1979	Kaminski	364/200
4,241,416	12/1980	Tarcty-Hornoch	364/900
4,429,368	1/1984	Kurii	364/200
4,495,563	1/1985	McDonough	364/200
4,511,960	4/1985	Boudreau	364/200
4,511,961	4/1985	Penton	364/200
4,571,677	2/1986	Hirayama	364/200
4,635,193	1/1987	Moyer	364/200
4,740,895	4/1988	Sargent	364/200
4,782,461	11/1988	Mick	364/900
4,811,345	3/1989	Johnson	364/200
4,819,234	4/1989	Huber	364/200
4,910,663	3/1990	Bailey	364/200

OTHER PUBLICATIONS

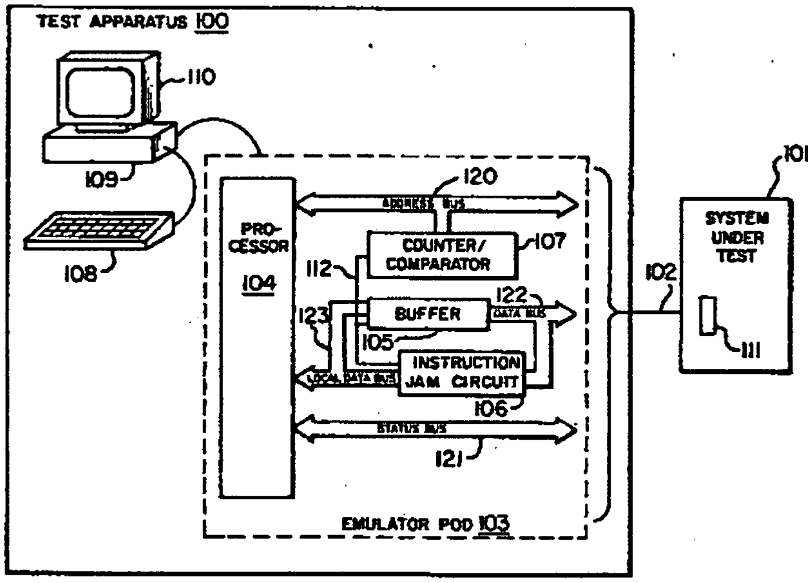
"Custom Trigger Chip Speeds 32-Bit Emulator To 33 MHz and Beyond"; Novellino; Electronic Design; Jan. 26, 1989; pp. 77 and 78.

Primary Examiner—Thomas C. Lee
Assistant Examiner—Eric Coleman

[57] **ABSTRACT**

The test apparatus for monitoring the operation of a processor that has multiple instruction fetch capability monitors the instruction memory to record the sequence of program instructions that are retrieved by the processor from program memory. The test apparatus determines when a jump operation is executed and determines the target of the jump oepration by inserting a break point instruction in place of one of the two program instructions that is retrieved by the processor from program memory. This instruction substitution is accomplished by an instruction jamming circuit that forces the break point instruction onto the processor data bus as part of the program instruction fetch cycle in lieu of one of the instruction retrieved as part of the execution of the jump instruction. If the break point operation is executed, then the target address of the jump operation is the address location that contains the break point instruction that was substituted for one of the program instructions retrieved from the instruction memory. In this case, the test apparatus responds to the execution of the break point instruction by replacing the program instruction originally retrieved from program memory and substituted for by the break point instruction. Thus, the break point instruction acts as a flag to indicate that this address is the target address of the jump instruction. If the break point instruction is not executed by the processor, it is because the jump instruction target address is the location that contains the other retrieved program instruction.

33 Claims, 2 Drawing Sheets



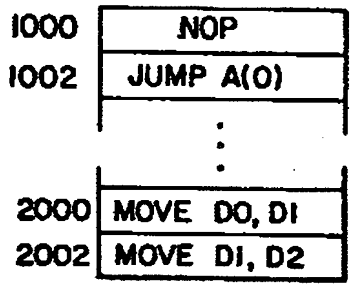

1000
NOP
1002
JUMP A(0)
2000
MOVE D0, D1
2002
MOVE D1, D2

FIG. 2.

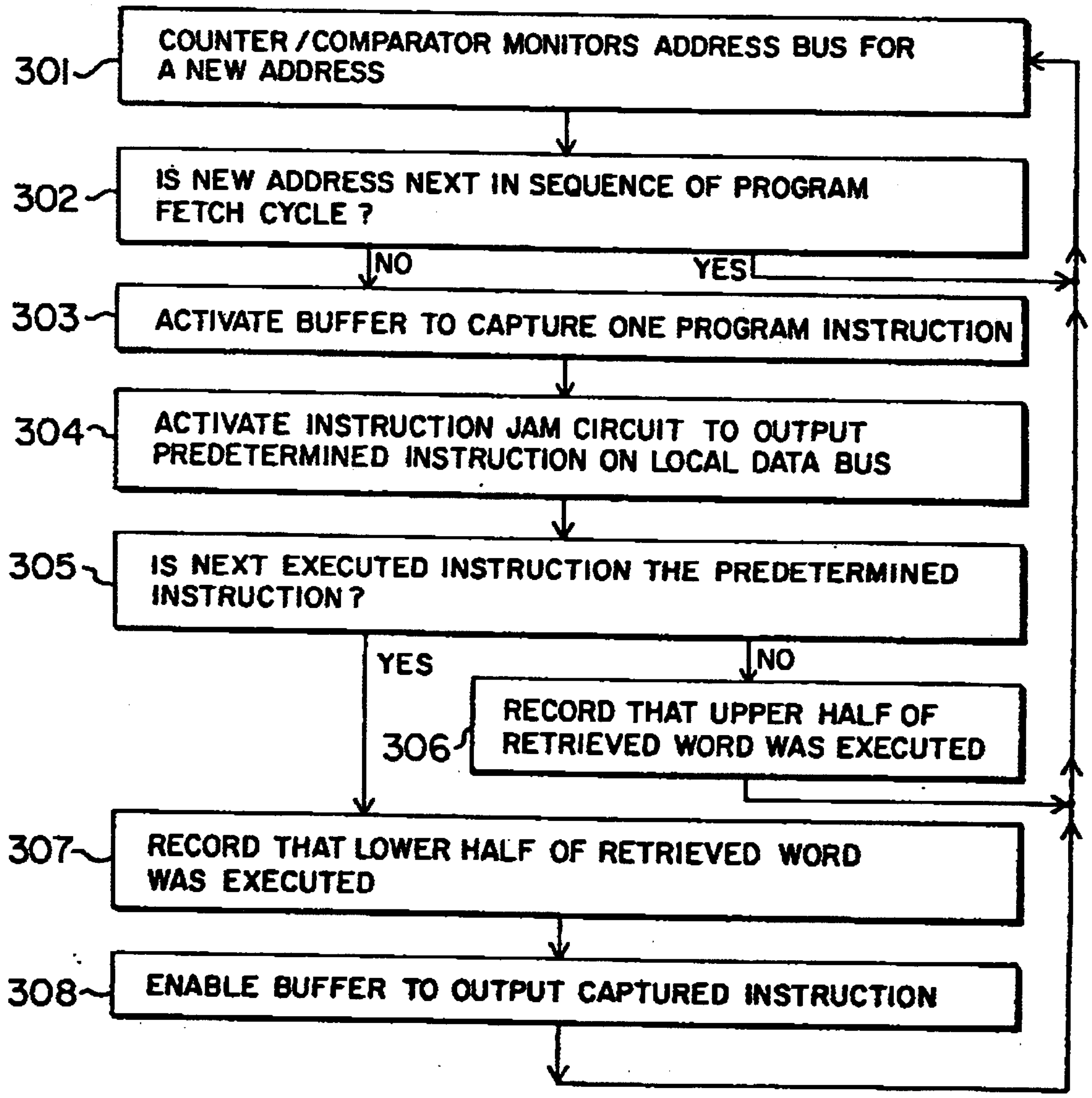

301 COUNTER/COMPARATOR MONITORS ADDRESS BUS FOR A NEW ADDRESS
302 IS NEW ADDRESS NEXT IN SEQUENCE OF PROGRAM FETCH CYCLE ?
NO
YES
303 ACTIVATE BUFFER TO CAPTURE ONE PROGRAM INSTRUCTION
304 ACTIVATE INSTRUCTION JAM CIRCUIT TO OUTPUT PREDETERMINED INSTRUCTION ON LOCAL DATA BUS
305 IS NEXT EXECUTED INSTRUCTION THE PREDETERMINED INSTRUCTION ?
YES
NO
306 RECORD THAT UPPER HALF OF RETRIEVED WORD WAS EXECUTED
307 RECORD THAT LOWER HALF OF RETRIEVED WORD WAS EXECUTED
308 ENABLE BUFFER TO OUTPUT CAPTURED INSTRUCTION

FIG. 3.

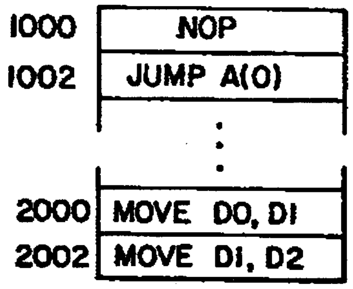

1000
NOP
1002
JUMP A(0)
2000
MOVE D0, D1
2002
MOVE D1, D2

FIG. 2.

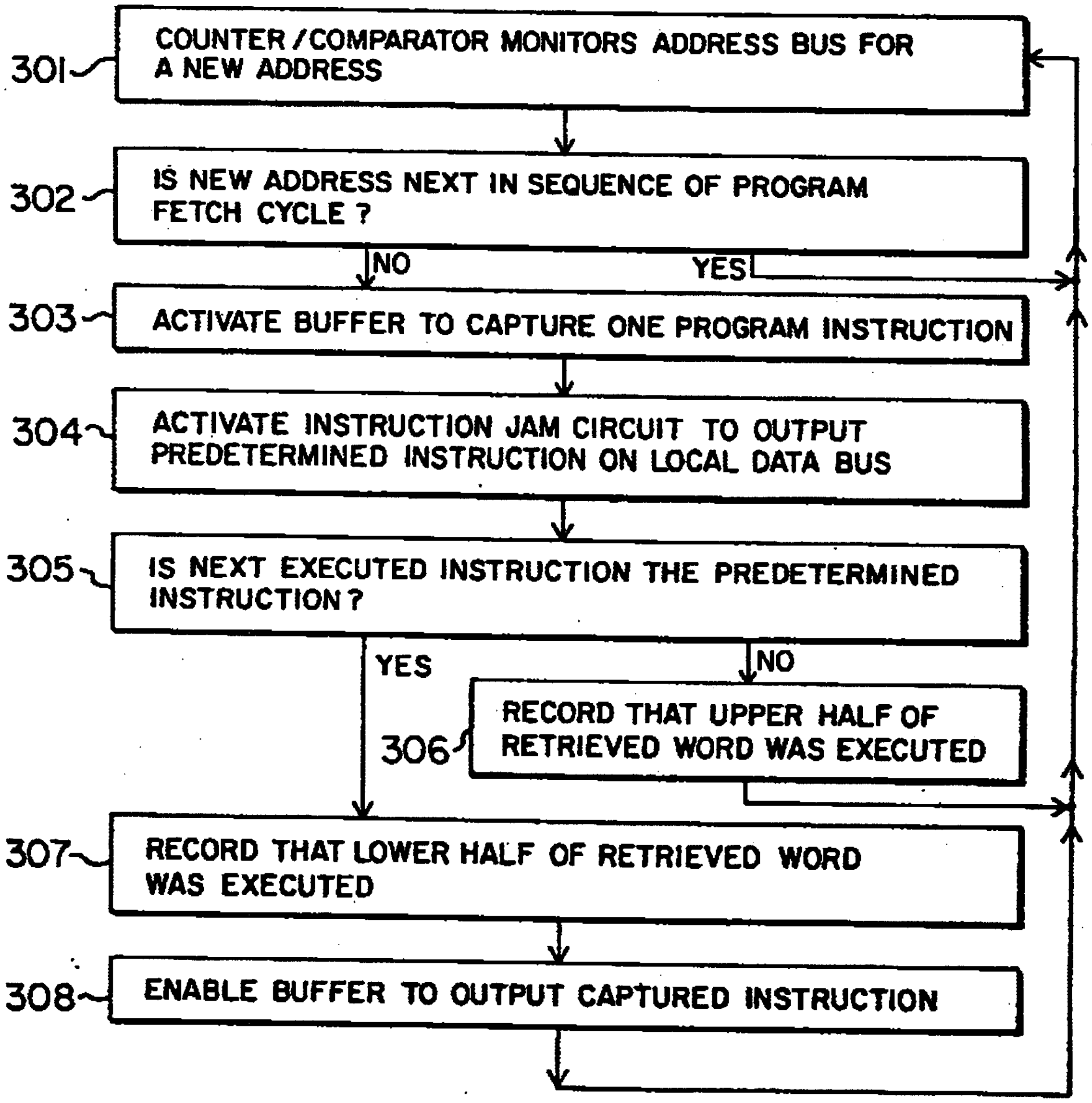

301 COUNTER / COMPARATOR MONITORS ADDRESS BUS FOR A NEW ADDRESS
302 IS NEW ADDRESS NEXT IN SEQUENCE OF PROGRAM FETCH CYCLE ?
NO
YES
303 ACTIVATE BUFFER TO CAPTURE ONE PROGRAM INSTRUCTION
304 ACTIVATE INSTRUCTION JAM CIRCUIT TO OUTPUT PREDETERMINED INSTRUCTION ON LOCAL DATA BUS
305 IS NEXT EXECUTED INSTRUCTION THE PREDETERMINED INSTRUCTION ?
YES
NO
306 RECORD THAT UPPER HALF OF RETRIEVED WORD WAS EXECUTED
307 RECORD THAT LOWER HALF OF RETRIEVED WORD WAS EXECUTED
308 ENABLE BUFFER TO OUTPUT CAPTURED INSTRUCTION

FIG. 3.

Boufarah et al.

[45] **Date of Patent: Jun. 30, 1992**

[54] **SYSTEM FOR REDUCING DELAY IN INSTRUCTION EXECUTION BY EXECUTING BRANCH INSTRUCTIONS IN SEPARATE PROCESSOR WHILE DISPATCHING SUBSEQUENT INSTRUCTIONS TO PRIMARY PROCESSOR**

[75] Inventors: **Edmond J. Boufarah**, Austin; **Gregory F. Groboski**, Cedar Park; **Chien-Chyun Lee**; **Charles R. Moore**, both of Ausin, all of Tex.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **297,784**

[22] Filed: **Jan. 13, 1989**

[51] Int. Cl.⁵ **G06F 9/38**

[52] U.S. Cl. **395/375**; 364/230.6; 364/231.8; 364/261.5; 364/263; 364/263.1; 364/271.2; 364/931.5; 364/938.1; 364/938.2; 364/948; 364/948.3; 364/DIG. 1; 395/800

[58] Field of Search ... 364/200 MS File, 900 MS File; 395/375

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,699,526	10/1972	Iskiyan et al.	340/172.5
4,210,960	7/1980	Borgeson et al.	364/200
4,373,180	2/1983	Linde	364/200
4,390,946	6/1983	Lane	364/200
4,551,798	11/1985	Horvath	364/200
4,679,141	7/1987	Pomerene et al.	364/200
4,742,451	5/1988	Bruckert et al.	364/200
4,742,453	5/1988	Shibuya	364/200
4,777,594	10/1988	Jones	364/200
4,791,557	12/1988	Angel et al.	364/200
4,853,840	8/1989	Shibuya	364/200
4,853,889	8/1989	Ditzel et al.	364/900
4,912,635	3/1990	Nishimukai et al.	364/200
4,974,154	11/1990	Matsuo	364/200
4,991,078	2/1991	Wilheim et al.	364/200
4,991,080	2/1991	Emma et al.	364/200
4,992,932	2/1991	Ohshima	364/200

OTHER PUBLICATIONS

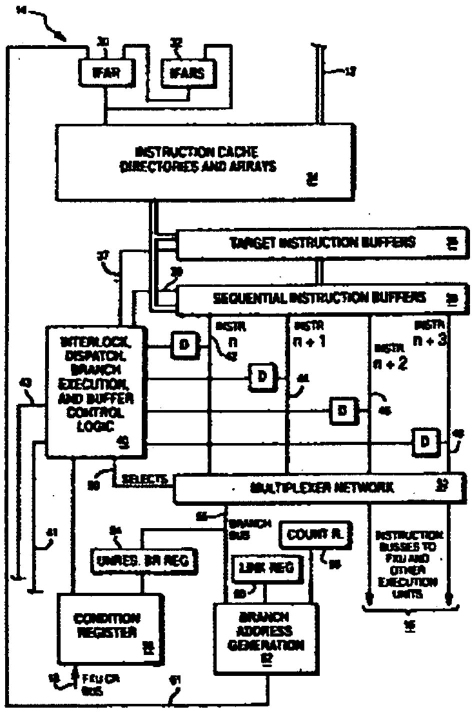
IBM Technical Disclosure Bulletin, vol. 29, No. 1, Jun., 1986, entitled "Branch-Processing Instruction Cache", pp. 357-359.

Primary Examiner—Thomas C. Lee
Assistant Examiner—Ken S. Kim
Attorney, Agent, or Firm—Thomas E. Tyson

[57] **ABSTRACT**

A data processing system including a circuit for storing a sequence of instructions, a circuit for determining if the instruction sequence includes a branch instruction, a circuit for storing a sequence of branch target instructions in response to the determination of the existence of a branch instruction in the stored sequence of instructions, a circuit for dispatching instructions in sequence after the branch instruction to a processor to be executed on condition that a branch is to be taken before a determination of whether said branch will be taken and simultaneously for determining if the branch is to be taken, any circuit for directing the processor to execute the instructions in sequence after the branch if the branch is not taken, or, if the branch is to be taken, for dispatching the branch target instruction sequence to the processor for execution.

32 Claims, 8 Drawing Sheets


IFAR
IFARS
INSTRUCTION CACHE DIRECTORIES AND ARRAYS
TARGET INSTRUCTION BUFFERS
SEQUENTIAL INSTRUCTION BUFFERS
INTERLOCK, DISPATCH, BRANCH EXECUTION, AND BUFFER CONTROL LOGIC
D
INSTR n
INSTR n + 1
INSTR n + 2
INSTR n + 3
SELECTS
MULTIPLEXER NETWORK
BRANCH BUS
COUNT R.
UNRES. BR REG
LINK REG
INSTRUCTION BUSSES TO FXU AND OTHER EXECUTION UNITS
CONDITION REGISTER
BRANCH ADDRESS GENERATION
FXU CR BUS

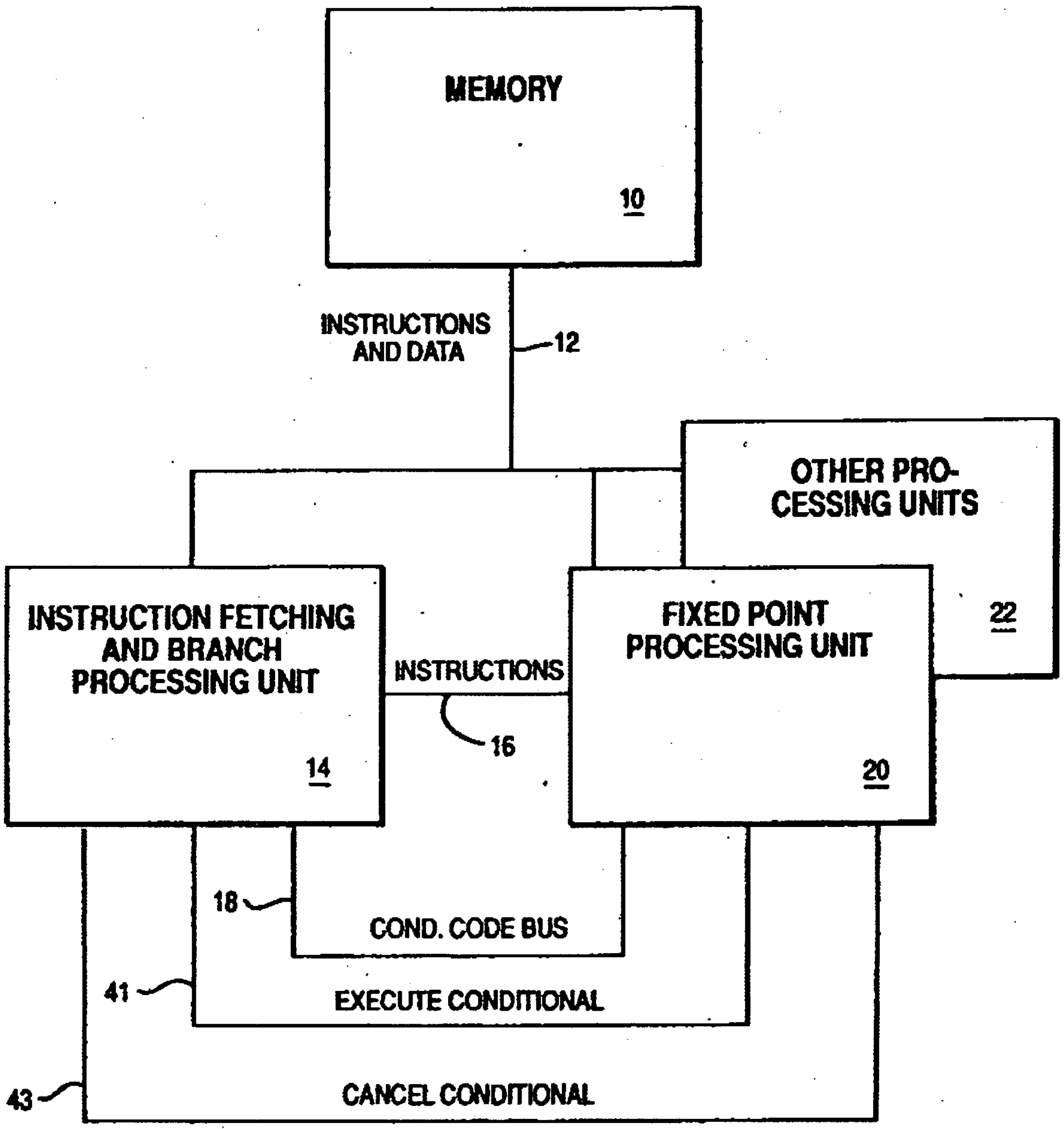
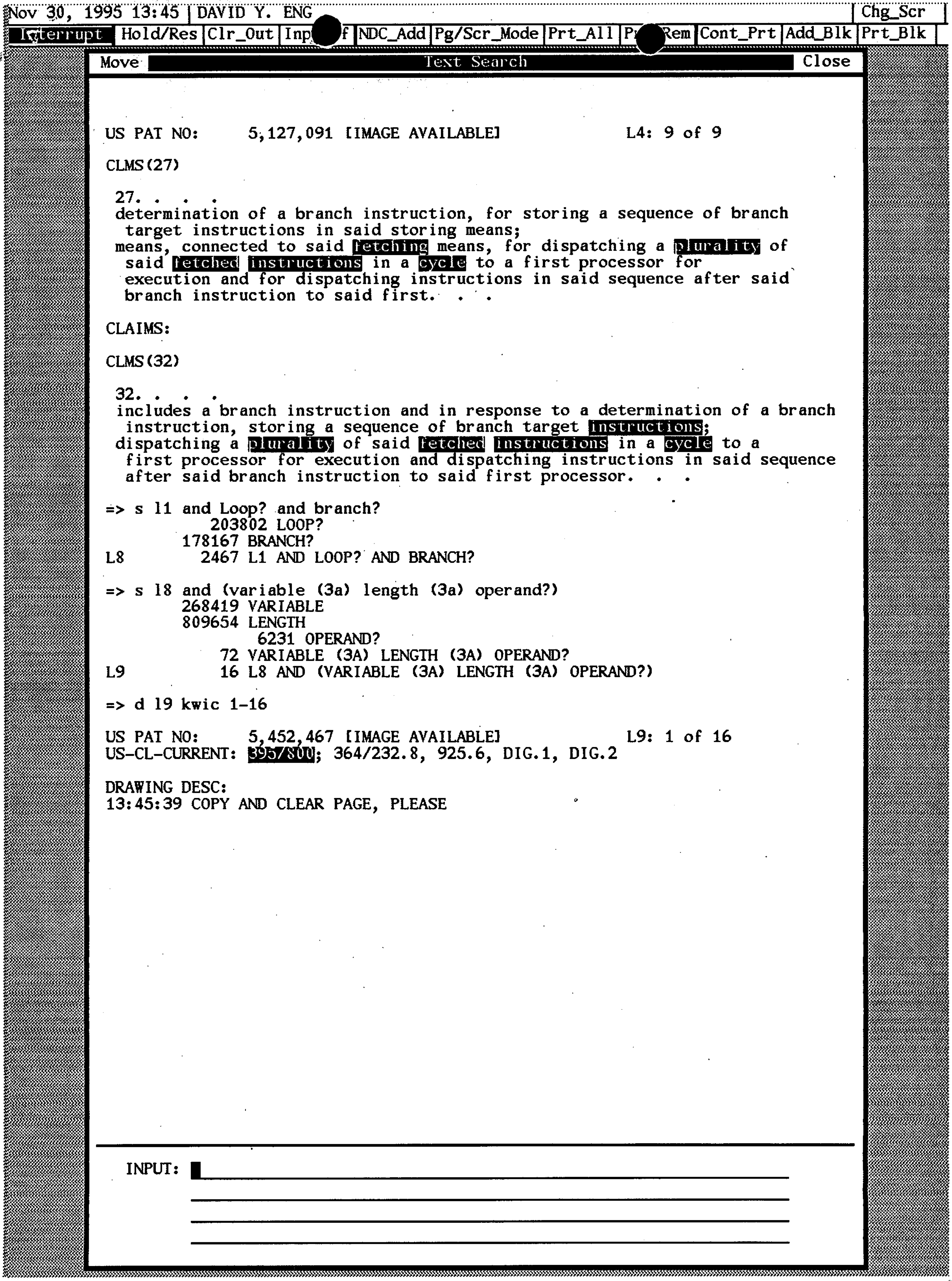
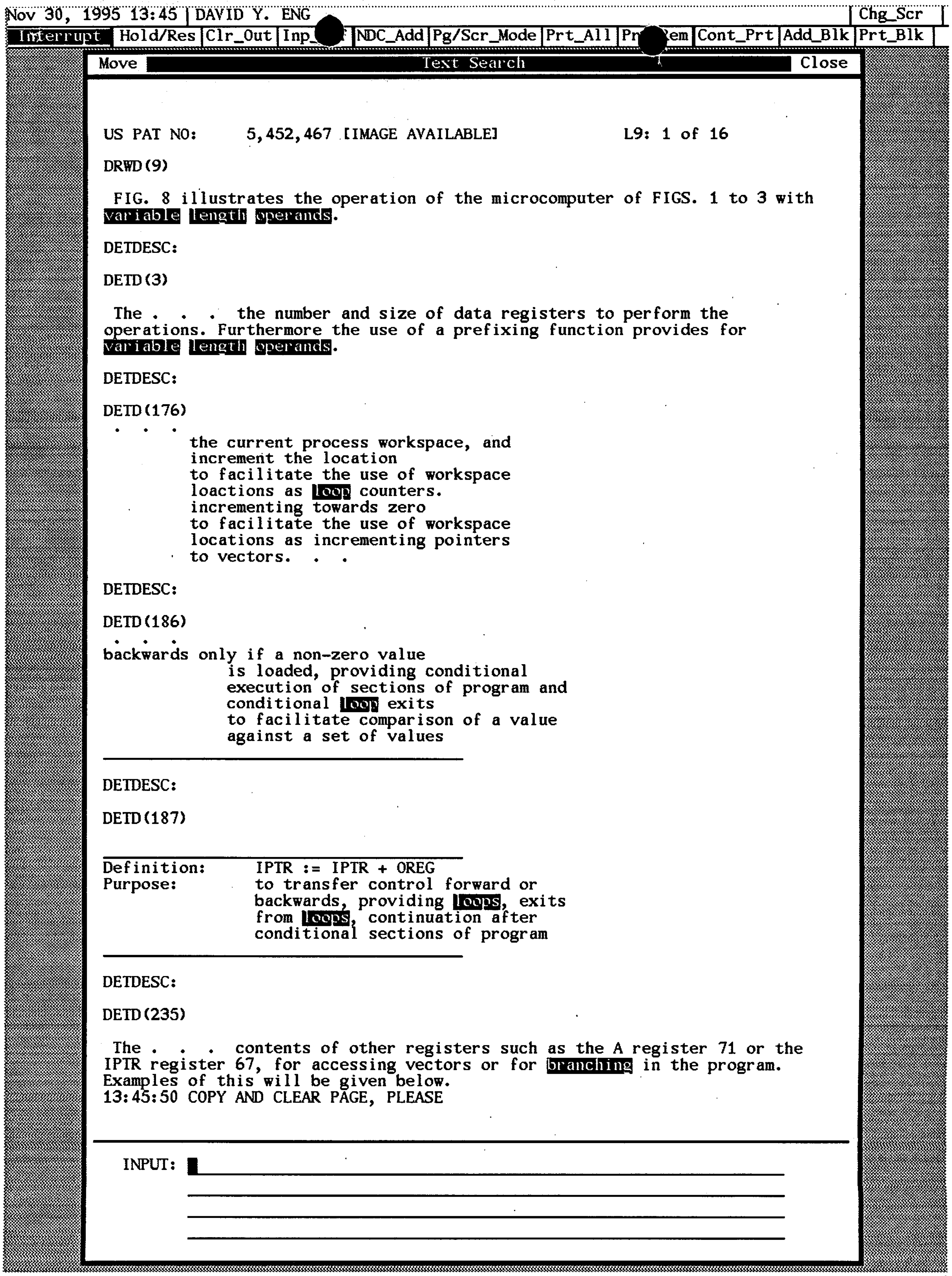

MEMORY
10
INSTRUCTIONS AND DATA
12
OTHER PROCESSING UNITS
22
INSTRUCTION FETCHING AND BRANCH PROCESSING UNIT
14
INSTRUCTIONS
16
FIXED POINT PROCESSING UNIT
20
18
COND. CODE BUS
41
EXECUTE CONDITIONAL
43
CANCEL CONDITIONAL

FIG. 1


Nov 30, 1995 13:45 | DAVID Y. ENG | Chg_Scr

```
US PAT NO:     5,127,091 [IMAGE AVAILABLE]             L4: 9 of 9

CLMS(27)

 27. . . .
 determination of a branch instruction, for storing a sequence of branch
  target instructions in said storing means;
 means, connected to said fetching means, for dispatching a plurality of
  said fetched instructions in a cycle to a first processor for
  execution and for dispatching instructions in said sequence after said
  branch instruction to said first. . .

CLAIMS:

CLMS(32)

 32. . . .
 includes a branch instruction and in response to a determination of a branch
  instruction, storing a sequence of branch target instructions;
 dispatching a plurality of said fetched instructions in a cycle to a
  first processor for execution and dispatching instructions in said sequence
  after said branch instruction to said first processor. . .

=> s l1 and Loop? and branch?
             203802 LOOP?
         178167 BRANCH?
L8         2467 L1 AND LOOP? AND BRANCH?

=> s l8 and (variable (3a) length (3a) operand?)
         268419 VARIABLE
         809654 LENGTH
                  6231 OPERAND?
              72 VARIABLE (3A) LENGTH (3A) OPERAND?
L9            16 L8 AND (VARIABLE (3A) LENGTH (3A) OPERAND?)

=> d l9 kwic 1-16

US PAT NO:     5,452,467 [IMAGE AVAILABLE]             L9: 1 of 16
US-CL-CURRENT: 395/800; 364/232.8, 925.6, DIG.1, DIG.2

DRAWING DESC:
13:45:39 COPY AND CLEAR PAGE, PLEASE
```

INPUT:



```
US PAT NO:      5,452,467 [IMAGE AVAILABLE]              L9: 1 of 16

DRWD(9)

 FIG. 8 illustrates the operation of the microcomputer of FIGS. 1 to 3 with
variable length operands.

DETDESC:

DETD(3)

 The . . .  the number and size of data registers to perform the
operations. Furthermore the use of a prefixing function provides for
variable length operands.

DETDESC:

DETD(176)
 . . .
         the current process workspace, and
         increment the location
         to facilitate the use of workspace
         loactions as loop counters.
         incrementing towards zero
         to facilitate the use of workspace
         locations as incrementing pointers
         to vectors.  . .

DETDESC:

DETD(186)
 . . .
backwards only if a non-zero value
             is loaded, providing conditional
             execution of sections of program and
             conditional loop exits
             to facilitate comparison of a value
             against a set of values
_______________________________________

DETDESC:

DETD(187)

_______________________________________
Definition:     IPTR := IPTR + OREG
Purpose:        to transfer control forward or
                backwards, providing loops, exits
                from loops, continuation after
                conditional sections of program
_______________________________________

DETDESC:

DETD(235)

 The . . .  contents of other registers such as the A register 71 or the
IPTR register 67, for accessing vectors or for branching in the program.
Examples of this will be given below.
13:45:50 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

```
Nov 30, 1995 13:45 | DAVID Y. ENG                                      | Chg_Scr
Interrupt | Hold/Res | Clr_Out | Inp_Off | NDC_Add | Pg/Scr_Mode | Prt_All | Prt_Rem | Cont_Prt | Add_Blk | Prt_Blk

Move                          Text Search                              Close

US PAT NO:      5,452,467 [IMAGE AVAILABLE]              L9: 1 of 16

DETD(235)

DETDESC:

DETD(240)

                    USE OF VARIABLE LENGTH OPERANDS

DETDESC:

DETD(241)

 As already explained above, the microcomputer is capable of operating with a
variable length operand. Although each instruction allocates 4 bit
locations to an operand, it is possible to build up in the operand register.
. .

DETDESC:

DETD(269)

 The "jump" function is used for branching in a program. The instruction to
be executed next by the processor for the current process is pointer to by.
. .

DETDESC:

DETD(275)

 The . . . prevent any single process from using the ALU 55 to the
exclusion of other processes. This operation is inserted into loops by the
compiler. This operation adds the current process to the end of the linked
list, stores the necessary pointers,. . .

DETDESC:

DETD(285)

 Line . . . added to facilitate explanation. Line 1 declares a variable to
exist; it is called "rotations". Line 2 is an endless loop because the
condition TRUE is always true. Start with zero rotations (line 4). Line 7
waits for any input from. . .

US PAT NO:      5,440,749 [IMAGE AVAILABLE]              L9: 2 of 16
US-CL-CURRENT:  395/800; 364/232.8, 244.3, 926.6, 931, 937.1, 965.4, DIG.1,
                  DIG.2

SUMMARY:

BSUM(17)

 In . . . instruction in the multiple instructions. In a modification of
this aspect of the invention, the microprocessor system additionally has a
loop counter connected to receive a decrement control signal from the means

  INPUT: █
```

US PAT NO: 5,440,749 [IMAGE AVAILABLE] L9: 2 of 16

BSUM(17)
In a further modification to this aspect of the. . .

DETDESC:

DETD(22)

A loop counter 92 is connected to a decrementer 94 by lines 96 and 98. The loop counter 92 is bidirectionally connected to the internal data bus 90 by line 100. Stack pointer 102, return stack pointer. . .

DETDESC:

DETD(78)
LOOP COUNTER 32 BITS

DETDESC:

DETD(88)

1. . . . a cache for 128 8-bit instructions. Therefore, the next instruction will almost always be already present in the cache. Long loops within the cache are also possible and more useful than the 4 instruction loops in the microprocessor 50.

DETDESC:

DETD(103)

7. The microprocessor 50 architecture offers two types of looping structures: LOOP-IF-DONE and MICRO-LOOP. The former takes an 8-bit to 24-bit operand to describe the entry point to the loop address. The latter performs a loop entirely within the 4 instruction queue and the loop entry point is implied as the first instruction in the queue. Loops entirely within the queue run without external instruction fetches and execute up to three times as fast as the long loop construct. The microprocessor 310 retains both constructs with a few differences. The microprocessor 310 microloop functions in the same fashion. . . 50 operation, except the queue is 1024-bits or 128 8-bit instructions long. The microprocessor 310 microloop can therefore contain jumps, branches, calls and immediate operations not possible in the 4 8-bit instruction microprocessor 50 queue.

DETDESC:

DETD(104)

Microloops . . . block move and compare functions. The larger microprocessor 310 queue allows entire digital signal processing or floating point algorithms to loop at high speed in the queue.

DETDESC:

DETD(107)

BRANCH
13:46:20 COPY AND CLEAR PAGE, PLEASE

INPUT:

```
Nov 30, 1995 13:45 | DAVID Y. ENG                                              | Chg_Scr
Interrupt | Hold/Res | Clr_Out | Inp_[illegible]f | NDC_Add | Pg/Scr_Mode | Prt_All | Pr[illegible]em | Cont_Prt | Add_Blk | Prt_Blk

Move                              Text Search                              Close

US PAT NO:     5,440,749 [IMAGE AVAILABLE]            L9: 2 of 16

DETD(107)

DETDESC:

DETD(108)

 BRANCH-IF-ZERO

DETDESC:

DETD(109)

 LOOP-IF-NOT-DONE

DETDESC:

DETD(110)
These instructions take a variable length address operand 8, 16 or 24
bits long. The microprocessor 50 next address logic treats the three operands
similarly by adding or.  .  .

DETDESC:

DETD(164)

 The .  .  .  often use it in place of the longer conditional JUMP
instruction. SKIP also makes possible microloops which exit when the loop
counts down or when the SKIP jumps to the next instruction group. The result
is very fast code.

DETDESC:

DETD(167)

 The .  .  .  repetitively from one to three instructions residing in the
instruction register 108. The microloop instruction works in conjunction with
the LOOP COUNTER 92 (FIG. 2) connected to the internal data bus 90. To
execute a microloop, the program stores a count in LOOP COUNTER 92.
MICROLOOP may be placed in the first, second, third, or last byte 420 of the
instruction register 108. If placed in the first position, execution will
just create a delay equal to the number stored in LOOP COUNTER 92 times the
machine cycle. If placed in the second, third, or last byte 420, when the
microloop instruction is executed, it will test the LOOP COUNT for zero. If
zero, execution will continue with the next instruction. If not zero, the
LOOP COUNTER 92 is decremented and the 2-bit microinstruction counter is
cleared, causing the preceding instructions in the instruction register to.
.  .

DETDESC:

DETD(204)

 The .  .  .  with a pipeline are keeping the pipe full with instructions.
Each time an out of sequence instruction such as a BRANCH or CALL occurs,
the pipe must be refilled with the next sequence. The resulting dead time to
refill the pipeline.  .  .
13:46:36 COPY AND CLEAR PAGE, PLEASE

 INPUT: █
```

Nov 30, 1995 13:48 | DAVID Y. ENG | Chg_Scr

Interrupt | Hold/Res | Clr_Out | Inp_ [illegible] | NDC_Add | Pg/Scr_Mode | Prt_All | Pr [illegible] em | Cont_Prt | Add_Blk | Prt_Blk

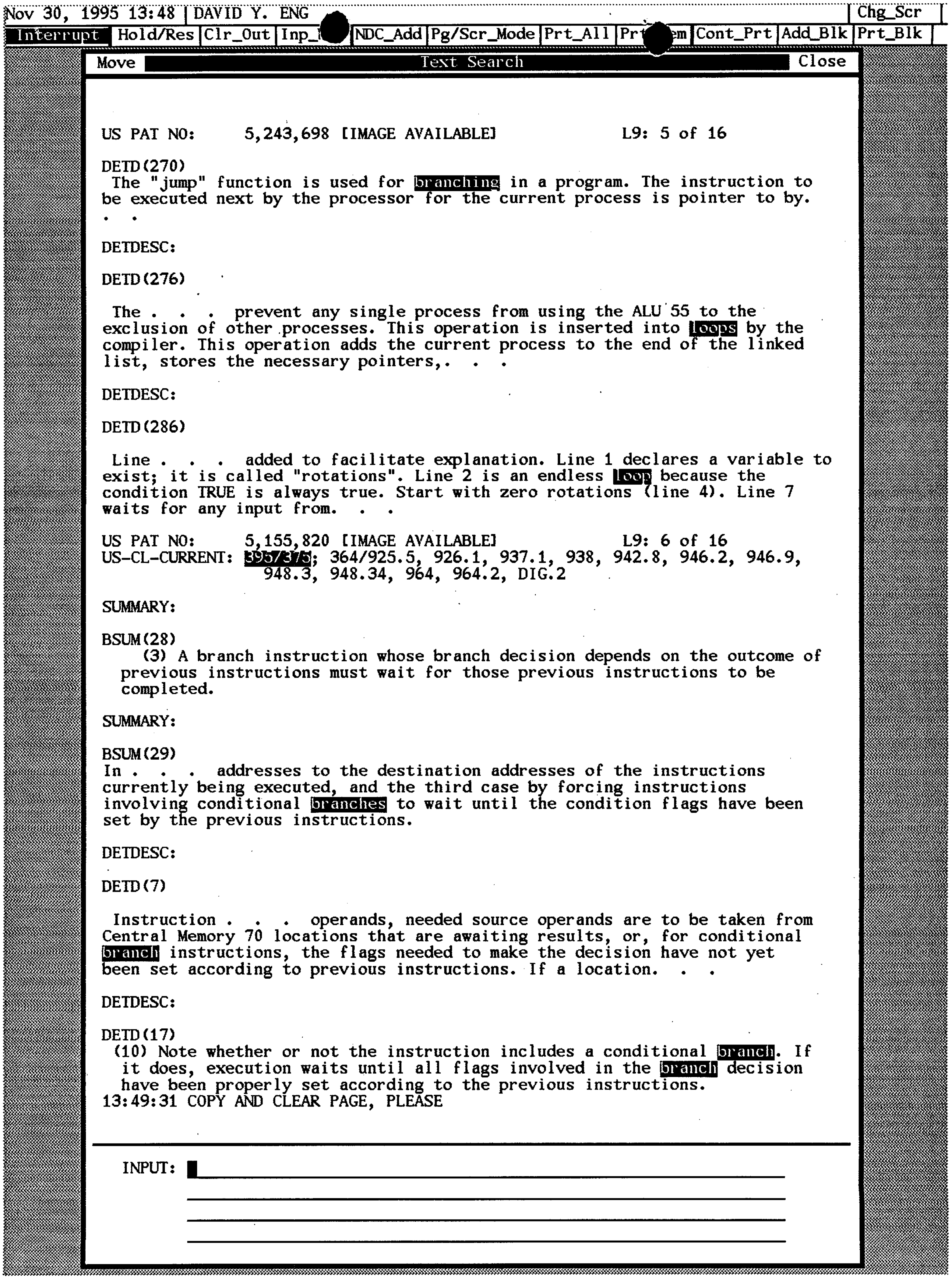
Move | Text Search | Close

```
US PAT NO:      5,321,823 [IMAGE AVAILABLE]              L9: 3 of 16

DETD(28)

DETDESC:

DETD(57)

 It . . . save-mask, reduces considerably the number of machine cycles
required to achieve the register save operation. In contrast, a microroutine
that loops through checking each bit of the save-mask and accumulating a
count of the bits set would take at least twelve. . .

DETDESC:

DETD(63)

 Concluding, . . . bits of the save-mask on the ABBus in just one step in
one machine cycle as opposed to a microcode loop which takes many steps.
This allows quick calculation of the memory requirement for saving the
registers (specified by the bit. . .

US PAT NO:      5,291,586 [IMAGE AVAILABLE]              L9: 4 of 16
US-CL-CURRENT: 395/500; 364/254.3, DIG.1; 395/375

SUMMARY:

BSUM(9)

 Another means of decoding the variable operand length field is by
table look-up as taught in IBM Technical Disclosure Bulletin, Volume 19, No.
1, dated June, 1976, by. . .

DETDESC:

DETD(18)

 In . . . 20-1b, a directory 20-1c for the pageable control store 20-1b, a
control store address register (CSAR) 20-1d, and an 8-element branch and
link (BAL STK) facility 20-1e. Machine state controls 20-2 include the global
controls 20-2a for the processor, an op branch table 20-2b connected to the
CSAR via the control store origin address bus which is used to generate the
initial. . . of a 370 condition code; a four-byte arithmetic logic unit
(ALU) 20-4c; an 8-byte rotate merge unit 20-4d; and a branch bit select
hardware 20-4e which allows the selection of bits from various registers
which determine the direction of a branch operation, the bits being
selected from general purpose registers, working registers, and the condition
registers. A floating point processor 20-5. . .

DETDESC:

DETD(20)

 A . . . and directory 20-3b. The hit lines are connected to the data
cache 18-2b; a generated instruction as an address, the op-branch table
providing the beginning address of the microcode routine needed to execute
the instruction. These instructions, as well as others, require more than 1
cycle to execute. Therefore, instruction decoding is suspended while the
op-branch table is being searched. In the case of microcode, the I-BUS is
utilized to provide microinstructions to the decoding hardware.. . .
13:48:22 COPY AND CLEAR PAGE, PLEASE
```

INPUT:

```
Nov 30, 1995 13:48 | DAVID Y. ENG                                          | Chg_Scr
Interrupt | Hold/Res | Clr_Out | Inp_ f | NDC_Add | Pg/Scr_Mode | Prt_All | Pr em | Cont_Prt | Add_Blk | Prt_Blk
Move                              Text Search                              Close

US PAT NO:      5,291,586 [IMAGE AVAILABLE]              L9: 4 of 16

DETD(20)

DETDESC:

DETD(34)

 The .  .  .  of operations required depends on the R fields of the
instruction. Furthermore, the operations can be implemented in a hardware
loop. In each loop the register fields are compared with the R3 field of
the instruction. The loop terminates when a match occurs.

DETDESC:

DETD(36)

 Referring .  .  .  the original macro-instruction is decoded from the
instruction register 20-4F and .vertline.R1 - R3.vertline.>1, the control
latch mini-mode 20-42 indicates loop mode or mini-instruction mode, is set.
The +0/+1/+2 incrementer 20-44 is seeded with the R1 field of the
macro-instruction and.  .  .

DETDESC:

DETD(37)

 The first storage address is formed by field D2(B2). However, when the
loop mode control latch 20-42 is set, the B2 and D2 fields of the
mini-instruction are ignored by the address generation.  .  .

US PAT NO:      5,243,698 [IMAGE AVAILABLE]              L9: 5 of 16
US-CL-CURRENT:  395/200.01; 364/241.7, 241.9, 923.5, 926.1, 927.92, 927.93,
                  927.94, 933, 940, 940.61, 940.62, 942, 949, 949.1, 950,
                  950.3, 957, 957.3, 959.1, 960, 965, 965.76, DIG.1, DIG.2;
                  395/800

DRAWING DESC:

DRWD(9)

 FIG. 8 illustrates the operation of the microcomputer of FIGS. 1 to 3 with
variable length operands.

DETDESC:

DETD(3)

 The .  .  .  the number and size of data registers to perform the
operations. Furthermore the use of a prefixing function provides for
variable length operands.

DETDESC:

DETD(177)
 .  .  .
            the current process workspace, and
            increment the location
            to facilitate the use of workspace
13:49:07 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

```
Nov 30, 1995 13:48 | DAVID Y. ENG                                          | Chg_Scr |
 Interrupt | Hold/Res | Clr_Out | Inp_    | NDC_Add | Pg/Scr_Mode | Prt_All | Pr   em | Cont_Prt | Add_Blk | Prt_Blk |

 Move                              Text Search                              Close

 US PAT NO:      5,243,698 [IMAGE AVAILABLE]              L9: 5 of 16

 DETD(177)
           locations as loop counters,
           incrementing towards zero
           to facilitate the use of workspace
           locations as incrementing pointers
           to vectors.  .  .

 DETDESC:

 DETD(187)

 ______________________________________
 Definition:   IPTR := IPTR + OREG
 Purpose:      to transfer control forward or
               backwards, providing loops, exits
               from loops, continuation after
               conditional sections of program
 ______________________________________

 DETDESC:

 DETD(189)
   .  .  .
 backwards only if a non-zero value
              is loaded, providing conditional
              execution of sections of program and
              conditional loop exits
              to facilitate comparison of a value
              against a set of values
 ______________________________________

 DETDESC:

 DETD(236)

  The .  .  .  contents of other registers such as the A register 71 or the
 IPTR register 67, for accessing vectors or for branching in the program.
 Examples of this will be given below.

 DETDESC:

 DETD(241)

                       USE OF VARIABLE LENGTH OPERANDS

 DETDESC:

 DETD(242)

  As already explained above, the microcomputer is capable of operating with a
 variable length operand. Although each instruction allocates 4 bit
 locations to an operand, it is possible to build up in the operand register.
 .  .

 DETDESC:

 DETD(270)

 13:49:18 COPY AND CLEAR PAGE, PLEASE

   INPUT: █
```



```
Nov 30, 1995 13:48 | DAVID Y. ENG                                          Chg_Scr
Interrupt  Hold/Res Clr_Out Inp_  NDC_Add Pg/Scr_Mode Prt_All Pr   em Cont_Prt Add_Blk Prt_Blk
Move                              Text Search                                   Close

US PAT NO:      5,243,698 [IMAGE AVAILABLE]              L9: 5 of 16

DETD(270)
 The "jump" function is used for branching in a program. The instruction to
be executed next by the processor for the current process is pointer to by.
.  .

DETDESC:

DETD(276)

 The .  .  .  prevent any single process from using the ALU 55 to the
exclusion of other processes. This operation is inserted into loops by the
compiler. This operation adds the current process to the end of the linked
list, stores the necessary pointers,.  .  .

DETDESC:

DETD(286)

 Line .  .  .  added to facilitate explanation. Line 1 declares a variable to
exist; it is called "rotations". Line 2 is an endless loop because the
condition TRUE is always true. Start with zero rotations (line 4). Line 7
waits for any input from.  .  .

US PAT NO:      5,155,820 [IMAGE AVAILABLE]              L9: 6 of 16
US-CL-CURRENT: 395/375; 364/925.5, 926.1, 937.1, 938, 942.8, 946.2, 946.9,
                   948.3, 948.34, 964, 964.2, DIG.2

SUMMARY:

BSUM(28)
     (3) A branch instruction whose branch decision depends on the outcome of
  previous instructions must wait for those previous instructions to be
  completed.

SUMMARY:

BSUM(29)
In .  .  .  addresses to the destination addresses of the instructions
currently being executed, and the third case by forcing instructions
involving conditional branches to wait until the condition flags have been
set by the previous instructions.

DETDESC:

DETD(7)

 Instruction .  .  .  operands, needed source operands are to be taken from
Central Memory 70 locations that are awaiting results, or, for conditional
branch instructions, the flags needed to make the decision have not yet
been set according to previous instructions. If a location.  .  .

DETDESC:

DETD(17)
 (10) Note whether or not the instruction includes a conditional branch. If
  it does, execution waits until all flags involved in the branch decision
  have been properly set according to the previous instructions.
13:49:31 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

US PAT NO: 5,155,820 [IMAGE AVAILABLE] L9: 6 of 16

DETD(17)

DETDESC:

DETD(49)

Some . . . Control Unit 30. When the limit is reached a signal which could be used by certain instructions that include conditional branches, such as loop instructions, would be sent to the Control Unit 30.

DETDESC:

DETD(50)

When variable length operands are being used, the length must be sent to the Input Logic 72 or Output Logic 71 when operands are. . .

DETDESC:

DETD(66)
The . . . signals on the FL 121 and have ceased holding the FR 122 signal inactive. An instruction that involves a conditional branch that depends on the flags would not be allowed to proceed to make the branch decision until the FR 122 signal is active. An alternative would be to have a flag ready line corresponding to. . .

DETDESC:

DETD(77)
Branch True (BT) 113 signal which is active when the current instruction

is an inside branch and the branch condition is true.

DETDESC:

DETD(78)
Outside Branch True (OBT) 112 signal which is active when the current instruction is an outside branch and the branch condition is true.

DETDESC:

DETD(79)
The BT 113 and OBT 112 lines permit two types of branches, referred to as inside and outside branches. Other lines could be added if more than two types of branches are needed for a design. The Instruction/status Bus 27 matches the definition of the code buffer given in Patent (A). . .

DETDESC:

DETD(84)

Location . . . autodecremented after each time they are accessed and locations 01 through 07 may be used for either counting (i.e., for looping--described later) or indirect addressing. When used for counting, the decrement amount is always 1. When used for indirect addressing the. .

INPUT:

Nov 30, 1995 13:48 | DAVID Y. ENG | Chg_Scr

Interrupt | Hold/Res | Clr_Out | Inp_[illegible] | NDC_Add | Pg/Scr_Mode | Prt_All | Prt[illegible]em | Cont_Prt | Add_Blk | Prt_Blk

Move | Text Search | Close

US PAT NO: 5,155,820 [IMAGE AVAILABLE] L9: 6 of 16

DETD(84)

DETDESC:

DETD(85)

An . . . and/or bits 15 through 8 and destination operands being indicated by bits 7 through 0. There are two types of branch instructions. There are inside branch instructions which have 8-bit branch addresses occupying bits 7 through 0 and outside branch instructions which have 16-bit branch addresses occupying bits 15 through 0.

DETDESC:

DETD(92)

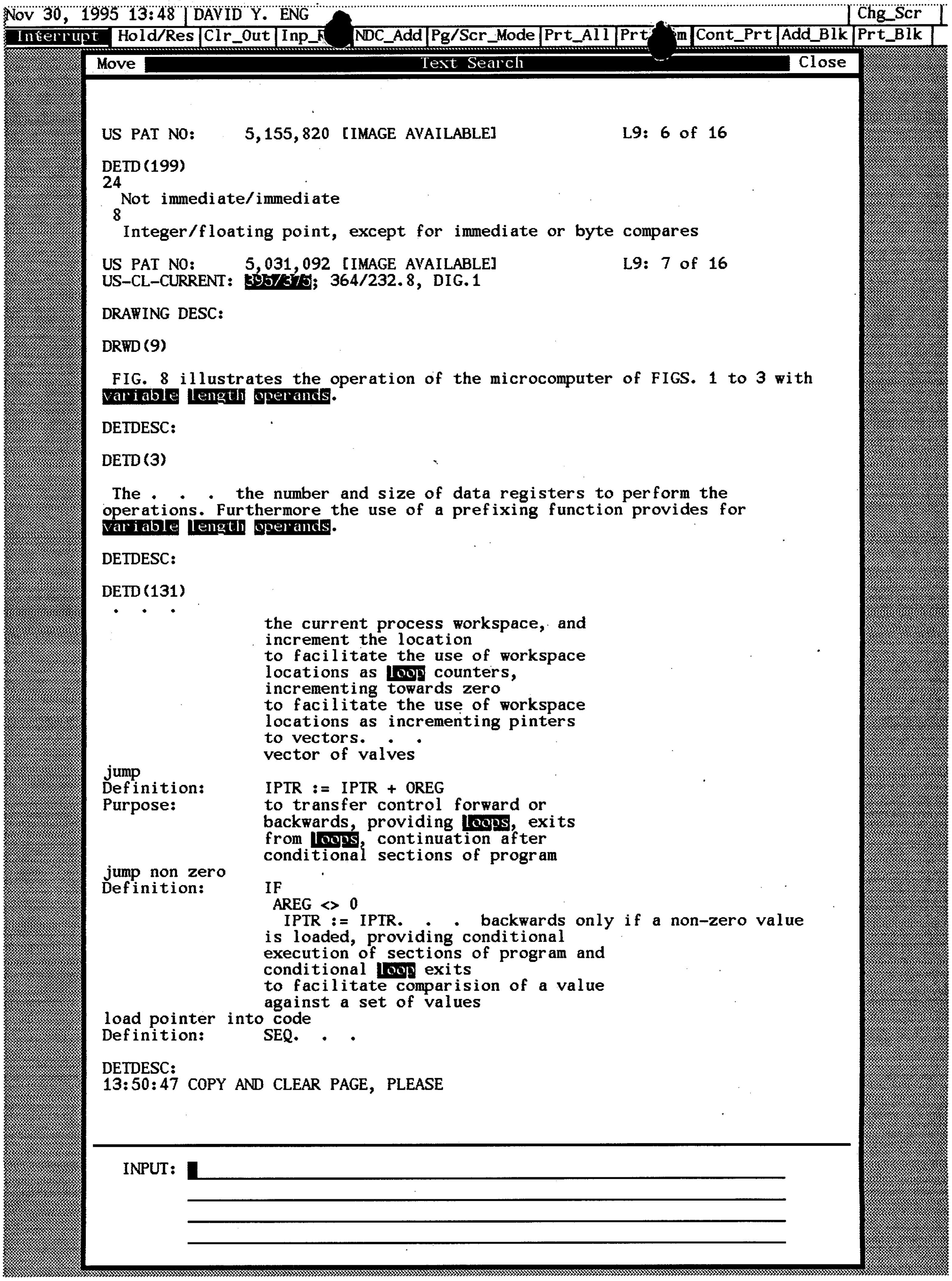
The present specific instruction set includes loop instructions and each loop instruction designates a location with an address in the range 01 through 07. The location is initially filled using a move/repeat instruction and each time the loop instruction is executed the contents of the location are tested. If they are 0 an active signal is sent from. . . Control Unit 30 over the CB 62 line that corresponds to the location. This signal being active causes the backward branch to not be taken; otherwise the backward branch is taken. After the contents of the location are tested they are decremented by 1.

DETDESC:

DETD(96)

```
________________________________________
Do not care = --
Data memory addresses = a (first source), b (second source),
  c (destination)
Branch address = d
Immediate operand = i
When a source operand may be immediate bit 24 = 0 indicates
it is not immediate and bit 24 = 1 indicates it is immediate.
Shift/rotate count = s
Branch condition = rrr
rrr      Direct     Complement
010      GE (.gtoreq.)
                    LT (<)        Unsigned
011      GE (.gtoreq.)
                    LT (<)        Signed. . . GT (>)    LE (.ltoreq.)
                                  Unsigned
101      GT (>)     LE (.ltoreq.)
                                  Signed
110      NE (.noteq.)
                    EQ (=)        Unsigned/signed
Branch direction = y
0 - forward
1 - backward
Flags branch condition = xxxxxxxx
00000000 - unconditional branch
00000001 - divide by zero
00000010 - exponent underflow
00000100 - exponent overflow
13:49:55 COPY AND CLEAR PAGE, PLEASE
```

INPUT:



```
US PAT NO:      5,155,820 [IMAGE AVAILABLE]              L9: 6 of 16

DETD(199)
24
  Not immediate/immediate
 8
  Integer/floating point, except for immediate or byte compares

US PAT NO:      5,031,092 [IMAGE AVAILABLE]              L9: 7 of 16
US-CL-CURRENT: 395/375; 364/232.8, DIG.1

DRAWING DESC:

DRWD(9)

 FIG. 8 illustrates the operation of the microcomputer of FIGS. 1 to 3 with
variable length operands.

DETDESC:

DETD(3)

 The .  .  .  the number and size of data registers to perform the
operations. Furthermore the use of a prefixing function provides for
variable length operands.

DETDESC:

DETD(131)
 .  .  .
                 the current process workspace, and
                 increment the location
                 to facilitate the use of workspace
                 locations as loop counters,
                 incrementing towards zero
                 to facilitate the use of workspace
                 locations as incrementing pinters
                 to vectors.  .  .
                 vector of valves
jump
Definition:      IPTR := IPTR + OREG
Purpose:         to transfer control forward or
                 backwards, providing loops, exits
                 from loops, continuation after
                 conditional sections of program
jump non zero
Definition:      IF
                  AREG <> 0
                   IPTR := IPTR.  .  .  backwards only if a non-zero value
                 is loaded, providing conditional
                 execution of sections of program and
                 conditional loop exits
                 to facilitate comparision of a value
                 against a set of values
load pointer into code
Definition:      SEQ.  .  .

DETDESC:
13:50:47 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

```
Nov 30, 1995 13:48 | DAVID Y. ENG                                          | Chg_Scr |
Interrupt | Hold/Res | Clr_Out | Inp_F  | NDC_Add | Pg/Scr_Mode | Prt_All | Prt  m | Cont_Prt | Add_Blk | Prt_Blk
Move                          Text Search                          Close


US PAT NO:      5,031,092 [IMAGE AVAILABLE]              L9: 7 of 16

DETD(134)

 The .  .  .  contents of other registers such as the A register 71 or the
IPTR register 67, for accessing vectors or for branching in the program.
Examples of this will be given below.

DETDESC:

DETD(139)

                      USE OF VARIABLE LENGTH OPERANDS

DETDESC:

DETD(140)

 As already explained above, the microcomputer is capable of operating with a
variable length operand. Although each instruction allocates 4 bit
locations to an operand, it is possible to build up in the operand register.
.  .

DETDESC:

DETD(168)

 The "jump" function is used for branching in a program. The instruction to
be executed next by the processor for the current process is pointer to by.
.  .

DETDESC:

DETD(174)

 The .  .  .  prevent any single process from using the ALU 55 to the
exclusion of other processes. This operation is inserted into loops by the
compiler. This operation adds the current process to the end of the linked
list, stores the necessary pointers,.  .  .

DETDESC:

DETD(192)

 Line .  .  .  added to facilitate explanation. Line 1 declares a variable to
exist; it is called "rotations". Line 2 is an endless loop because the
condition TRUE is always true. Start with zero rotations (line 4). Line 7
waits for any input from.  .  .

US PAT NO:      4,967,326 [IMAGE AVAILABLE]              L9: 8 of 16
US-CL-CURRENT: 395/800; 364/229, 229.1, 230, 230.3, 231.4, 231.6, 232.8,
                 242.94, 262.4, 262.81, 271, 271.2, 271.3, 281.3, 281.4,
                 281.8, DIG.1; 395/280

DRAWING DESC:

DRWD(9)

 FIG. 8 illustrates the operation of the microcomputer of FIGS. 1 to 3 with
13:51:01 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

[illegible]

```
Nov.30, 1995 13:51 | DAVID Y. ENG                                              | Chg_Scr
Interrupt | Hold/Res | Clr_Out | Inp_  | NDC_Add | Pg/Scr_Mode | Prt_All | Prt   m | Cont_Prt | Add_Blk | Prt_Blk
Move                              Text Search                              Close


US PAT NO:     4,967,326 [IMAGE AVAILABLE]              L9: 8 of 16

DRWD(9)
variable length operands.

DETDESC:

DETD(3)

 The . . . the number and size of data registers to perform the
operations. Furthermore the use of a prefixing function provides for
variable length operands.

DETDESC:

DETD(130)
 . . .
in the current process
              workspace, and increment the location
              to facilitate the use of workspace
              locations as loop counters,
              incrementing towards zero
              to facilitate the use of workspace
              locations as incrementing pointers
              to vectors. . .
jump (function code 8)
Definition:
              IPTR := IPTR + OREG
Purpose:      to transfer control forward or
              backwards, providing loops, exits
              from loops, continuation after
              conditional sections of program
jump non zero (function code 9)
Definition:
              IF
              AREG<> 0
. . .      backwards only if a non-zero value
              is loaded, providing conditional
              execution of sections of program and
              conditional loop exits
              to facilitate comparison of a value
              against a set of values
load pointer into code (function code. . .

DETDESC:

DETD(133)

 The . . . contents of other registers such as the A register 71 or the
IPTR register 67, for accessing vectors or for branching in the program.
Examples of this will be given below.

DETDESC:

DETD(138)

                     USE OF VARIABLE LENGTH OPERANDS

DETDESC:
13:51:13 COPY AND CLEAR PAGE, PLEASE

  INPUT:
```

```
Nov 30, 1995 13:51 | DAVID Y. ENG                                          | Chg_Scr |
Interrupt | Hold/Res | Clr_Out | Inp_R[illegible]NDC_Add | Pg/Scr_Mode | Prt_All | Prt[illegible]m | Cont_Prt | Add_Blk | Prt_Blk |
Move                              Text Search                              Close

US PAT NO:     4,967,326 [IMAGE AVAILABLE]              L9: 8 of 16

DETD(139)

 As already explained above, the microcomputer is capable of operating with a
variable length operand. Although each instruction allocates 4 bit
locations to an operand, it is possible to build up in the operand register.
.  .

DETDESC:

DETD(168)

 The "jump" function (function code 8) is used for branching in a program.
The instruction to be executed next by the processor for the current process
is pointed to by.  .  .

DETDESC:

DETD(174)

 The .  .  .  prevent any single process from using the ALU 55 to the
exclusion of other processes. This operation is inserted into loops by the
compiler. This operation adds the current process to the end of the linked
list, stores the necessary pointers,.  .  .

DETDESC:

DETD(184)
Line .  .  .  added to facilitate explanation. Line 1 declares a variable to
exist; it is called "rotations". Line 2 is an endless loop because the
condition TRUE is always true. Start with zero rotations (line 4). Line 7
waits for any input from.  .  .

US PAT NO:     4,819,151 [IMAGE AVAILABLE]              L9: 9 of 16
US-CL-CURRENT: 395/650; 364/231, 231.9, 238.5, 239, 239.4, 240, 240.2,
                 241.9, 242.94, 244, 244.6, 254, 254.6, 254.9, 255.1, 255.5,
                 255.7, 262, 262.1, 262.4, 262.8, 271, 271.3, 271.5, 280,
                 281.3, 281.8, 284, 284.4, DIG.1

DRAWING DESC:

DRWD(9)

 FIG. 8 illustrates the operation of the microcomputer of FIGS. 1 to 3 with
variable length operands.

DETDESC:

DETD(3)

 The .  .  .  the number and size of data registers to perform the
operations. Furthermore the use of a prefixing function provides for
variable length operands.

DETDESC:

DETD(144)
13:51:32 COPY AND CLEAR PAGE, PLEASE

   INPUT: _
```

```
Nov. 30, 1995 13:51 | DAVID Y. ENG                                  | Chg_Scr |
Interrupt | Hold/Res | Clr_Out | Inp_I | NDC_Add | Pg/Scr_Mode | Prt_All | Prt em | Cont_Prt | Add_Blk | Prt_Blk
Move                              Text Search                              Close

US PAT NO:     4,819,151 [IMAGE AVAILABLE]              L9: 9 of 16

DETD(144)
 . . .
in the current process
         workspace, and increment the location
         to facilitate the use of workspace
         locations as loop counters,
         incrementing towards zero
         to facilitate the use of workspace
         locations as incrementing pointers
         to vectors. . .
jump (function code 8)
Definition:
         IPTR: = IPTR + OREG
Purpose: to transfer control forward or
         backwards, providing loops, exits
         from loops, continuation after
         conditional sections of program
jump non zero (function code 9)
Definition:
         IF
         AREG <> 0.  .  .  backwards only if a non-zero value
         is loaded, providing conditional
         execution of sections of program and
         conditional loop exits
         to facilitate comparison of a value
         against a set of values
load pointer into code (function code.  .  .

DETDESC:

DETD(146)

 The . . . contents of other registers such as the A register 71 or the
IPTR register 67, for accessing vectors or for branching in the program.
Examples of this will be given below.

DETDESC:

DETD(150)

                    USE OF VARIABLE LENGTH OPERANDS

DETDESC:

DETD(151)

 As already explained above, the microcomputer is capable of operating with a
variable length operand. Although each instruction allocates 4 bit
locations to an operand, it is possible to build up in the operand register.
. .

DETDESC:

DETD(180)

 The "jump" function (function code 8) is used for branching in a program.
The instruction to be executed next by the processor for the current process
is pointed to by. . .
13:51:42 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

```
Nov.30, 1995 13:51 | DAVID Y. ENG                                                     | Chg_Scr |
Interrupt | Hold/Res | Clr_Out | Inp_R | NDC_Add | Pg/Scr_Mode | Prt_All | Prt m | Cont_Prt | Add_Blk | Prt_Blk

Move                                  Text Search                                        Close


US PAT NO:      4,819,151 [IMAGE AVAILABLE]                L9: 9 of 16

DETD(180)

DETDESC:

DETD(186)

 The . . . prevent any single process from using the ALU 55 to the
exclusion of other processes. This operation is inserted into loops by the
compiler. This operation adds the current process to the end of the linked
list, stores the necessary pointers,. . .

DETDESC:

DETD(196)

 Line . . . added to facilitate explanation. Line 1 declares a variable to
exist; it is called "rotations". Line 2 is an endless loop because the
condition TRUE is always true. Start with zero rotations (line 4). Line 7
waits for any input from. . .

US PAT NO:      4,785,393 [IMAGE AVAILABLE]                L9: 10 of 16
US-CL-CURRENT: 395/375; 364/232.8, 238, 243, 244, 244.6, 252, 258, 258.1,
                 258.2, 258.3, 259, 259.5, 259.7, 260.4, 260.8, 263, 716,
                 736, DIG.1

SUMMARY:

BSUM(2)

 Typically, . . . issue addresses that are applied to the microprogram
memory to access the microinstruction sequence to be executed. When
processing conditional branching instructions, the sequencer, or
microprogram controller, tests selected status bits of the data processor,
such as overflow, zero, carry or. . .

DRAWING DESC:

DRWD(7)

 FIG. 5A shows the format of a variable-length bit-field operand
instruction executed by the data processor of the present invention and its
accompanying width and position information.

DRAWING DESC:

DRWD(8)

 FIG. 5B shows the format of a variable-length bit-field operand
processed by the device of the present invention.

DETDESC:

DETD(11)

 Briefly, . . . which must be executed to perform the required function
(specified by the instruction held by the instruction register 68). A
branch to this address occurs through the microprogram sequence controller
13:51:57 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

```
Nov 30, 1995 13:51 | DAVID Y. ENG                                     | Chg_Scr |
Interrupt | Hold/Res | Clr_Out | Inp_J   NDC_Add | Pg/Scr_Mode | Prt_All | Prt   m | Cont_Prt | Add_Blk | Prt_Blk |
   Move                         Text Search                           Close

US PAT NO:     4,785,393 [IMAGE AVAILABLE]             L9: 10 of 16

DETD(11)
12. The "machine" instruction may call for several microinstructions to be.
. .

DETDESC:

DETD(78)

                 2. Variable-Length Bit Field Operand Instructions

DETDESC:

DETD(84)

 The . . . the variable-length bit field instruction format and the
instruction bit assignments, and FIG. 5B illustrates the bit assignments for
the variable-length bit field operand to which the variable-length
bit field instructions apply. Depending on bits I.sub.8 and I.sub.7 of the
variable-length bit field instruction, accompanying the instruction and. .
.

DETDESC:

DETD(85)

                 TABLE VI
______________________________________
Variable-Length Operand Parameter Source
Inst
Bits  Width Source      Position Source
I.sub.8
    I.sub.7
          W.sub.4 -W.sub.0 Term.
                      Status Reg.
                             P.sub.5. . .

DETDESC:

DETD(140)

 The . . . handled internally by the processor 24. The only requirements
of the sequence controller 12 are that it be capable of looping and
conditional branching. Each of the three types of byte-boundary-aligned
division operations will be described in reference to a flowcharted
algorithm, comprising FIG.. . .

DETDESC:

DETD(142)

 In . . . lags the calculation of the quotient bit by one step, a
correction step may be required upon exit from the loop implementing the
successive reduction of the remainder. If the resulting remainder is not
exactly zero, it should be the same. . .

DETDESC:
13:52:14 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

```
Nov 30, 1995 13:53 | DAVID Y. ENG                                              | Chg_Scr
Interrupt | Hold/Res | Clr_Out | Inp_R  NDC_Add | Pg/Scr_Mode | Prt_All | Prt_ m | Cont_Prt | Add_Blk | Prt_Blk
Move                              Text Search                              Close

=> d his

     (FILE 'USPAT' ENTERED AT 13:29:43 ON 30 NOV 95)
                SET PAGELENGTH 62
                SET LINELENGTH 78
L1        26679 S 395/?/CCLS
L2            5 S L1 AND (FETCH? (5A) MULTIPLE (5A) INSTRUCTIONS (5A) CYCLE)
L3            4 S L1 AND (FETCH? (5A) PLURALITY (5A) INSTRUCTIONS (5A) CYCLE)
L4            9 S L3 OR L2
L5            3 S L4 AND BRANCH/AB
L6            0 S L5 AND LOOP?/AB
L7            2 S L5 AND LOOP?
L8         2467 S L1 AND LOOP? AND BRANCH?
L9           16 S L8 AND (VARIABLE (3A) LENGTH (3A) OPERAND?)

=> d kwic 19 16

US PAT NO:      3,654,448 [IMAGE AVAILABLE]             L9: 16 of 16
TITLE:          INSTRUCTION EXECUTION AND RE-EXECUTION WITH IN-LINE BRANCH
                  SEQUENCES
US-CL-CURRENT:  395/182.15; 364/243, 243.5, 251, 251.1, 251.2, 252.3, 252.6,
                  254, 254.5, 255.1, 255.2, 255.4, 261.3, 261.5, 262, 262.2,
                  262.4, 262.8, 265, 265.4, 265.6, 266.3, 266.5, 267, 267.3,
                  DIG.1

ABSTRACT:
Instruction re-execution for error recovery is enhanced by function skipping
conditional branches in the execution control sequence. Branches
continuing the main execution stream control execution and re-execution of
basic functions and ancillary verification and saving functions required for.
.  .  signals, 'scratchpad' saving of the particular function result signals
in fast access general purpose buffer storage and setting of re-execution
branch conditions to signal for skipping of basic functions when
re-executing instructions after basic function result signals have been
saved. Function skipping branches are taken therefor only in late
re-execution; i.e. only when the re-execution branch condition for function
skipping has been set prior to occurrence of error in execution. In the
function skipping branch previously saved basic function result signals
required for continued execution are obtained directly from fast access
buffer storage, thereby eliminating.  .  .  function. This is especially
useful as the associated operand signals may no longer be available at
re-execution time. The re-execution branch condition for function skipping
is reset at conclusion of each instruction execution control sequence.
Foregoing control organization has the advantage.  .  .  function results
need not be reprocessed and recalculated during error recovery. Also by
virtue of the standardized organization of the branch the controls may be
adapted piecemeal to a variety of different system recovery functions with
minimal cost/performance degradation.

SUMMARY:

BSUM(8)

 A .  .  .  design into discrete non-overlapping and non-dependent function
segments. The re-execution sequence is permitted to skip over successfully
completed segments by branching. For later consideration it is noted that
the last mentioned application also discloses a condition trigger which is
13:53:50 COPY AND CLEAR PAGE, PLEASE

  INPUT: █
```

```
Nov 30, 1995 13:55 | DAVID Y. ENG                                          | Chg_Scr
Interrupt | Hold/Res | Clr_Out | Inp_K  NDC_Add | Pg/Scr_Mode | Prt_All | Prt   | Cont_Prt | Add_Blk | Prt_Blk

Move                              Text Search                              Close

functions in re-executions which follow late error (i.e. error after RCT has
been set). Results of Add Loop operations are saved in fast access buffers
discussed later.

DETDESC:

DETD(28

=> d l9 kwic 15

US PAT NO:      3,739,352 [IMAGE AVAILABLE]              L9: 15 of 16
US-CL-CURRENT:  395/421.04; 364/238.4, 240.1, 243, 243.3, 243.7, 245, 245.1,
                  251, 251.1, 251.3, 252.3, 252.6, 254.9, 255.1, 255.5, 258,
                  258.1, 259, 259.3, 259.5, 260, 260.2, 261.3, 261.9, 262.4,
                  262.8, DIG.1

SUMMARY:

BSUM(2)

 This invention relates to digital processors, and more particularly, is
concerned with processing operands and data of variable field length.

SUMMARY:

BSUM(7)

 The . . . string is repeated until the width specified by the control
register is reduced to zero, in which case the processor branches out of
the string to fetch the next program instruction.

DETDESC:

DETD(7)

 Control . . . microoperator in the M-register. The A-register 32 can be
set to any new address from the data bus to permit branching to a different
string of microoperators in the M-string memory 28. A stack memory 34 is
preferably provided which operates. . .

DETDESC:

DETD(14)

 Referring again to FIG. 3, the next microoperator in the list is the
Branch microoperator. This microoperator is identified by 110 in the top 3
bits of the word in the M-register 30 and. . . being transferred to the 13
least significant bit positions of the A-register 32. This causes the
M-string memory 28 to branch to a new location, for example, to loop back
to the start of a string of microoperators or to jump to a different string
of microoperators in the. . .

DETDESC:

DETD(23)

 By this arrangement, the next microoperator following the Bias microoperator
in the program string might be a Branch operation. The Branch operation
is executed only if CPL is set to 0 by the Bias operation, otherwise the
Branch operation is replaced by a NoOP, and the next microoperator in the
string following the Branch is fetched into the M-register 30.
13:54:54 COPY AND CLEAR PAGE, PLEASE

  INPUT:
```